

计算机学院专业课程

计算机组成

流水线处理器 形式建模综合方法

高小鹏

北京航空航天大学计算机学院

挑战

- 流水线设计目标：给定指令集，确保基于该指令集的任意程序均能全速运行
 - ◆ 正确：能发现所有可能导致冒险的指令组合
 - ◆ 性能：尽力转发～～凡能转发的，绝不暂停！
 - 用暂停来规避转发，背离了流水线设计初衷
- 挑战：通过枚举构造冲突的方法，难以证明穷尽所有的可能
 - ◆ 箴言：测试只能发现错误，不能证明没有错误！
- 方法：针对任意指令集，正确、高效的构造全速转发流水线

TIPS

方法适用于单发射标准流水线，不适用乱序执行等最先进流水线

流水线方法的概述

- 基本思路：因为是全速流水线，所以转发是暂停的前提，

	数据通路	控制
转发	构造无转发的基础流水线	与单周期控制器相同
	增加转发电路	
暂停	根据流水线架构构造每条指令的 T_{use} 和 T_{new}	
	构造暂停/转发策略矩阵（覆盖性分析）	
	根据策略矩阵生成暂停控制表达式	
	根据策略矩阵生成转发控制表达式	

- 数据通路构造方法
 - ◆ 基础流水线规划
 - ◆ 建模指令RTL
 - ◆ RTL制导的独立数据通路
 - ◆ 综合无转发数据通路
 - ◆ 综合转发电路
 - ◆ 构造功能MUX控制表达式
- 暂停及转发的分析方法
- 暂停机制构造方法
- 转发机制构造方法
- 控制冒险处理机制

功能部件规划

- PC、RF：作为两个端点。PC为发起，RF为结束

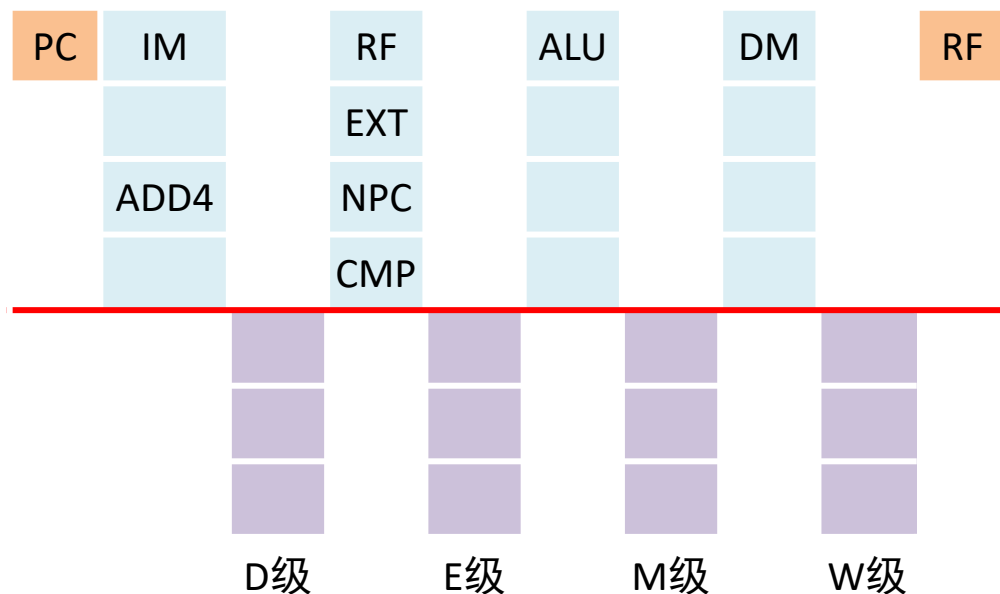
- F级：IM、ADD4

- D级：RF、EXT、NPC、CMP

- E级：ALU

- M级：DM

- W级：无



流水线寄存器命名

TIPS

望文生义：内容与形式统一

■ 指导思想：使用无二义性的名字；直观易懂

◆ 例如rt就存在二义性



• 可能代表第2个源寄存器编号，也可能是写寄存器编号

名字	宽度	描述	对应的指令域	命名考虑
A1	5位	第1个源寄存器编号	当前只有rs	与RF设计一致 A3需要由rd/rt/+31转换得到（后续介绍）
A2	5位	第2个源寄存器编号	当前只有rt	
A3	5位	目的寄存器编号	rd或rt或+31	
V1	32位	RF的第1个寄存器输出值		Value的首字母
V2	32位	RF的第2个寄存器输出值		
E32	32位	EXT的32位扩展结果		扩展的英文缩写
AO	32位	ALU计算结果		沿用多周期命名
DR	32位	DM输出值		
PC4	32位	下一条指令地址		



流水线数据通路描述表

		指令
PC		
IM		
ADD4		
D级	IR	
	PC4	
RF	A1	
	A2	
EXT		
NPC	PC4	
	I26	
CMP	D1	
	D2	
E级	V1	
	V2	
	A1	
	A2	
	A3	
	EXT	
	PC4	
ALU	A	
	B	
M级	V2	
	A2	
	AO	
	A3	
	PC4	
DM	A	
	WD	
W级	A3	
	PC4	
	AO	
	DR	
RF	A3	
	WD	

- ❑ 1、描述表的架构：与流水线执行路径尽量保持一致
- ❑ 2、RF：将读出与写入分离，更易于理解5阶段
- ❑ 3、流水寄存器在连接时有2种情况
 - ◆ 1) 保存刚产生的信息：与前级功能部件密切相关
 - 例如：E级的A3 
 - ◆ 2) 级间传递信息：简单的前后级衔接
 - 例如：M级和W级的A3 
- ❑ 4、由3可知，大量流水寄存器的连接关系是固定的，因此问题焦点是解决信息的**最初来源**
 - ◆ 例如A3：E级初值来自IR，M级/W级则为简单传递
 - ◆ 例如AO：M级初值来自ALU，W级则为简单传递

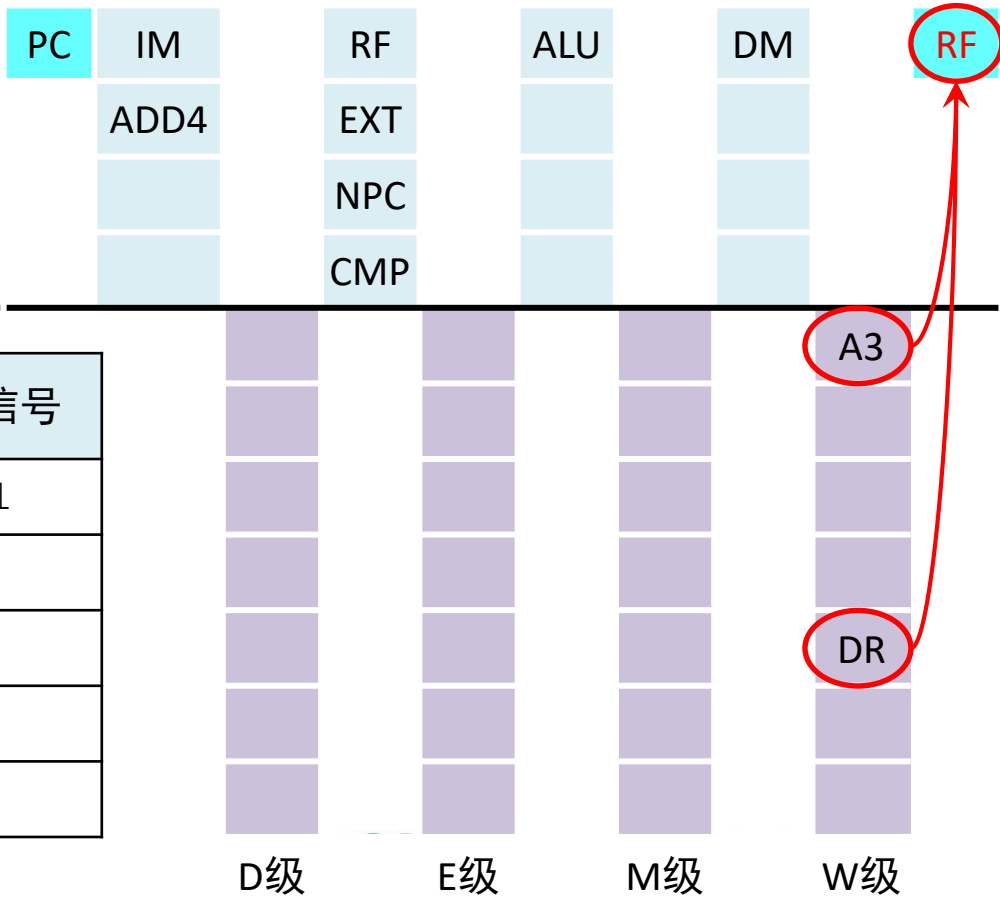


- 数据通路构造方法
 - ◆ 基础流水线规划
 - ◆ 建模指令RTL
 - ◆ RTL制导的独立数据通路
 - ◆ 综合无转发数据通路
 - ◆ 综合转发电路
 - ◆ 构造功能MUX控制表达式
- 暂停及转发的分析方法
- 暂停机制构造方法
- 转发机制构造方法
- 控制冒险处理机制

建模lw的RTL：回写

$$R[rt] \leftarrow \text{MEM}[R[rs] + \text{sign_ext}(\text{imm16})]$$

- 以下展示如何采用反推方式建模lw的RTL
- Cycle -1: 因为DM的数据最终存储在DR中，因此只能是DR写入RF
 - 为了写入RF，则必须满足如下条件：
 - 提供目的寄存器编号，即A3
 - RF的写使能为1

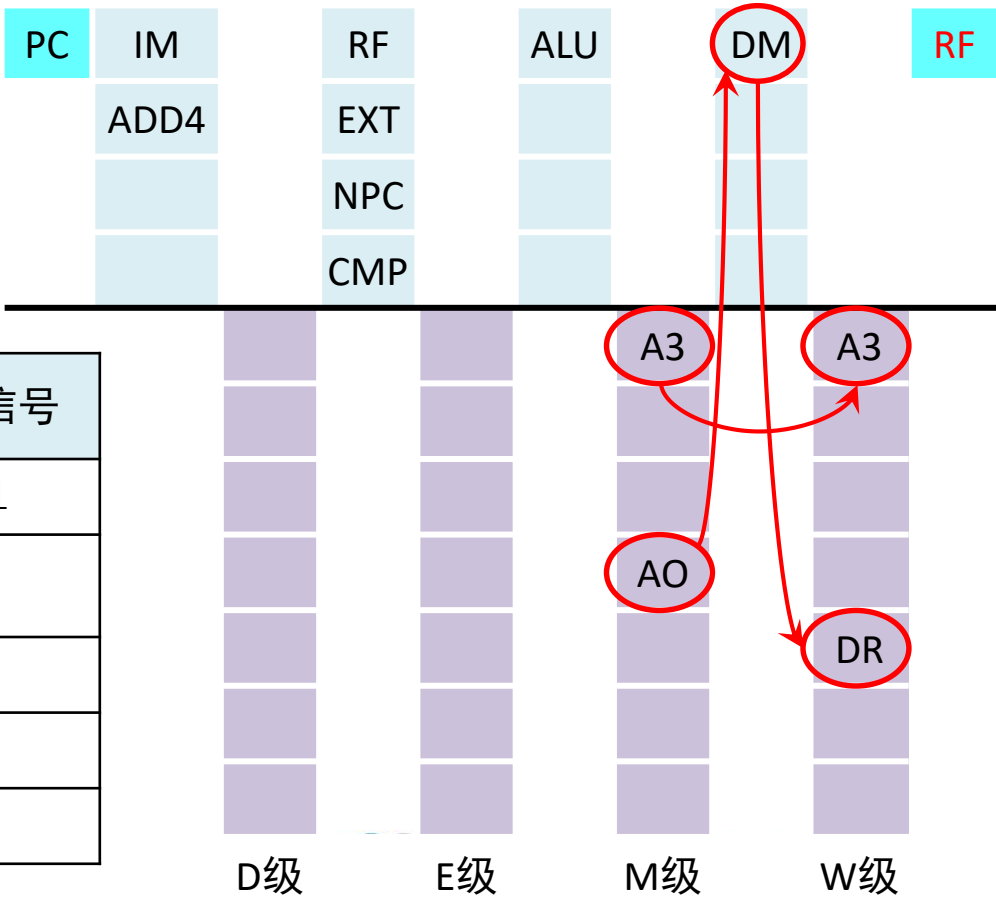


步骤	RTL	控制信号
回写	$RF[A3@W] \leftarrow DR@W$	$RFWr:1$

建模lw的RTL：访存

$$R[rt] \leftarrow \text{MEM}[R[rs] + \text{sign_ext}(\text{imm16})]$$

- Cycle -2: 从DM读出数据并写入DR
 - ◆ 读取DM，必须为DM提供地址，即AO
 - ◆ 注意：W级的A3只能依赖于M级的A3，因此必须从M级向W级传递A3

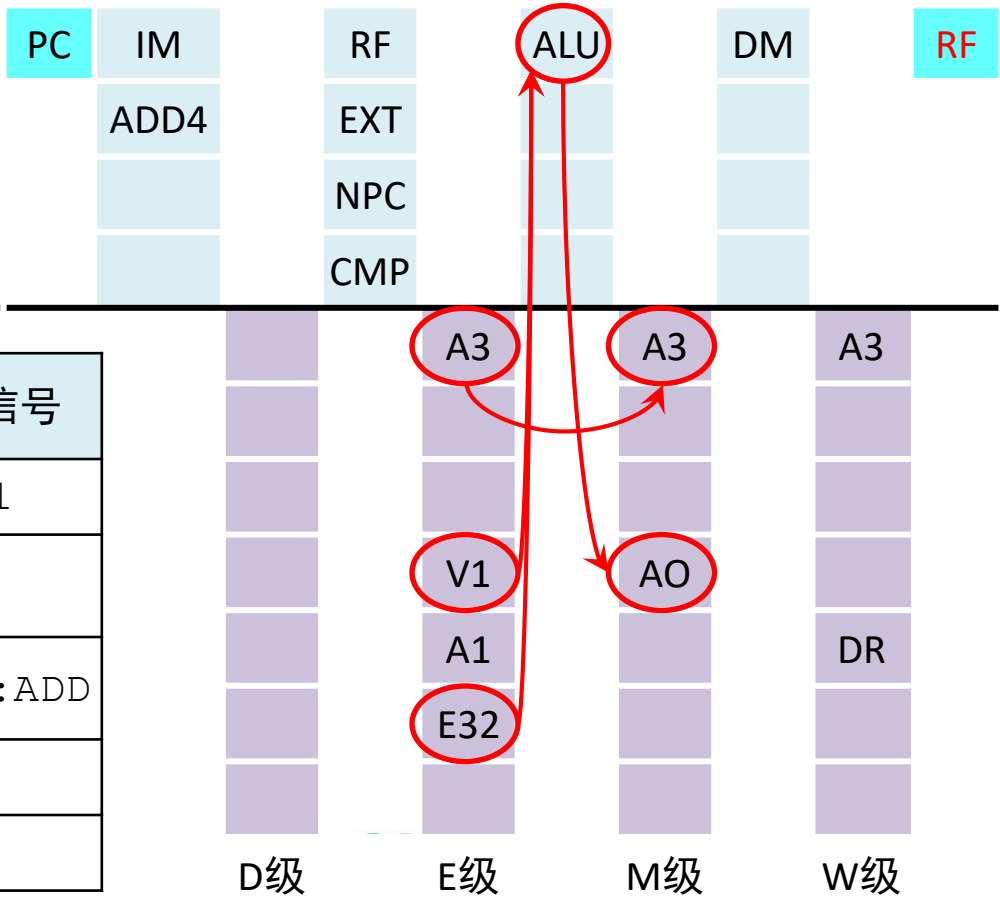


步骤	RTL	控制信号
回写	$RF[A3@W] \leftarrow DR@W$	$RFWr:1$
访存	$DR@W \leftarrow DM[AO@M]$ $A3@W \leftarrow A3@M$	

建模lw的RTL： 计算

$R[rt] \leftarrow MEM[R[rs] + sign_ext(imm16)]$

- Cycle -3: 基地址加偏移, 结果写入AO
 - 基地址来源于rs寄存器, 即V1; 偏移来源于扩展单元, 即E32
 - 由于有V1, 就必须同时建立A1
 - 控制器要判断是否与其他指令在A1(即rs)存在数据相关
 - A3仍然需要从E级向W级传递



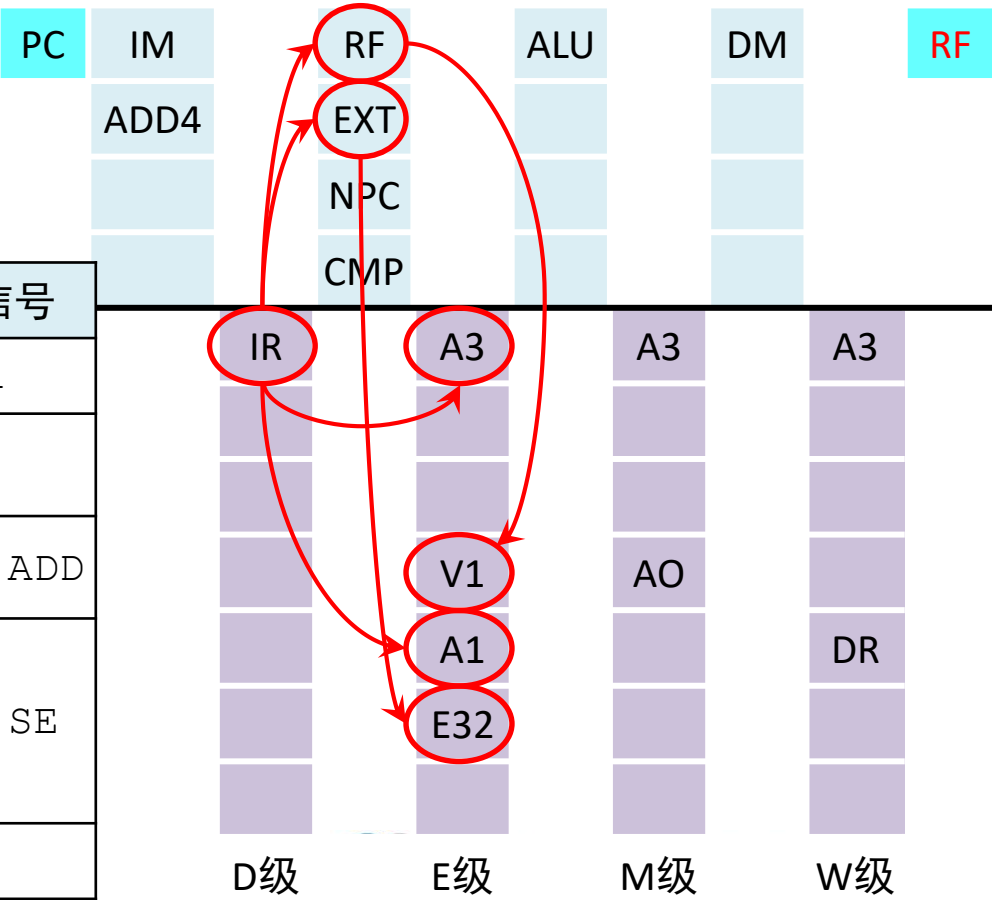
步骤	RTL	控制信号
回写	$RF[A3@W] \leftarrow DR@W$	$RFWr:1$
访存	$DR@W \leftarrow DM[AO@M]$ $A3@W \leftarrow A3@M$	
计算	$AO@M \leftarrow ALU(V1@E, E32@E)$ $A3@M \leftarrow A3@E$	$ALUOp:ADD$

建模lw的RTL：读指令

$R[rt] \leftarrow \text{MEM}[R[rs] + \text{sign_ext}(\text{imm16})]$

- Cycle -5: 读取rs(基地址)写入V1; 符号扩展结果写入E32(偏移)
 - 注意1: A3只能通过IR的rt域产生了
 - 注意2: 由于有V1, 就必须同时建立A1
 - A1的用途: 用于判断是否会与其他指令存在数据相关

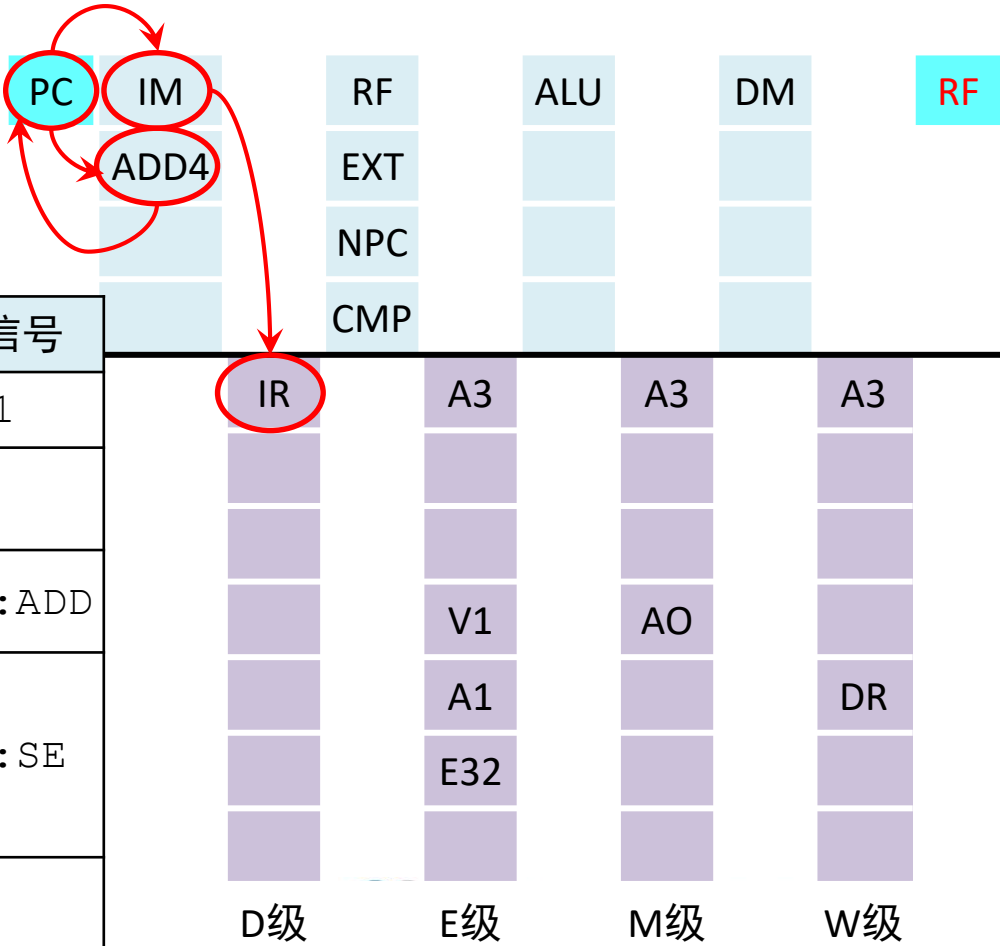
步骤	RTL	控制信号
回写	$RF[A3@W] \leftarrow DR@W$	$RFWr:1$
访存	$DR@W \leftarrow DM[AO@M]$ $A3@W \leftarrow A3@M$	
计算	$AO@M \leftarrow \text{ALU}(V1@E, E32@E)$ $A3@M \leftarrow A3@E$	$ALUOp:ADD$
读数	$V1@E \leftarrow RF[IR[rs]@D]$ $E32@E \leftarrow \text{EXT}(IR[i16]@D)$ $A3@E \leftarrow IR[rt]@D$ $A1@E \leftarrow IR[rs]@D$	$EXTOp:SE$



建模lw的RTL：取指令

$R[rt] \leftarrow \text{MEM}[R[rs] + \text{sign_ext}(\text{imm16})]$

- Cycle -5: 读IM并写入IR; PC指向下条指令
 - 注意: 除非暂停, 否则流水线每周期都取指令
 - 结论: PC.En(即PCWr)有效是常态
 - PC.En建模变为: 何时无效
 - IR: 同理处理



步骤	RTL	控制信号
回写	$RF[A3@W] \leftarrow DR@W$	$RFWr:1$
访存	$DR@W \leftarrow DM[AO@M]$ $A3@W \leftarrow A3@M$	
计算	$AO@M \leftarrow ALU(V1@E, E32@E)$ $A3@M \leftarrow A3@E$	$ALUOp:ADD$
读数	$V1@E \leftarrow RF[IR[rs]@D]$ $E32@E \leftarrow EXT(IR[i16]@D)$ $A3@E \leftarrow IR[rt]@D$ $A1@E \leftarrow IR[rs]@D$	$EXTOp:SE$
取值	$IR@D \leftarrow IM[PC]$ $PC \leftarrow ADD4(PC)$	

- 数据通路构造方法
 - ◆ 基础流水线规划
 - ◆ 建模指令RTL
 - ◆ RTL制导的独立数据通路
 - ◆ 综合无转发数据通路
 - ◆ 综合转发电路
 - ◆ 构造功能MUX控制表达式
- 暂停及转发的分析方法
- 暂停机制构造方法
- 转发机制构造方法
- 控制冒险处理机制

根据RTL建立数据通路表

基本思路：根据RTL描述，建立流水线寄存器、功能部件间的连接关系

以取指和计算两阶段为例

步骤	RTL	控制信号
取指	$IR@D \leftarrow IM[PC]$ $PC \leftarrow ADD4(PC)$	
读数	$V1@E \leftarrow RF[IR[rs]@D]$ $E32@E \leftarrow EXT(IR[i16]@D)$ $A3@E \leftarrow IR[rt]@D$ $V1@E \leftarrow IR[rs]@D$	EXTOp:SE
计算	$AO@M \leftarrow ALU(V1@E, E32@E)$ $A3@M \leftarrow A3@E$	ALUOp:ADD
访存	$DR@W \leftarrow DM[AO@M]$ $A3@W \leftarrow A3@M$	
回写	$RF[A3@W] \leftarrow DR@W$	RFWr:1

		Iw
	PC	ADD4
	IM	PC
	ADD4	PC
D级	IR	IM
	NPC	
RF	A1	IR[rs]@D
	A2	
EXT		IR[i16]@D
NPC	PC4	
	I26	
CMP	D1	
	D2	
E级	V1	RF.RD1
	V2	
	A1	IR[rs]@D
	A2	
	A3	IR[rt]@D
	E32	EXT
	PC4	
ALU	A	V1@E
	B	E32@E
M级	V2	
	A2	
	AO	ALU
	A3	A3@E
	PC4	
DM	A	AO@M
	WD	
W级	A3	A3@M
	PC4	
	AO	
	DR	DM
RF	A3	A3@W
	WD	DR@W



- 数据通路构造方法
 - ◆ 基础流水线规划
 - ◆ 建模指令RTL
 - ◆ RTL制导的独立数据通路
 - ◆ 综合无转发数据通路
 - ◆ 综合转发电路
 - ◆ 构造功能MUX控制表达式
- 暂停及转发的分析方法
- 暂停机制构造方法
- 转发机制构造方法
- 控制冒险处理机制

■ 每条指令
对应的数
据通路

		lw	sw	add	sub	ori	beq	j	jal	jr
PC		ADD4	ADD4	ADD4	ADD4	ADD4	ADD4 NPC	NPC	NPC	RF.RD1
IM		PC	PC	PC	PC	PC	PC	PC	PC	PC
ADD4		PC	PC	PC	PC	PC	PC	PC	PC	
D级	IR	IM	IM	IM	IM	IM	IM	IM	IM	IM
	NPC	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	
RF	A1	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D			IR[rs]@D
	A2		IR[rt]@D	IR[rt]@D	IR[rt]@D		IR[rt]@D			
EXT		IR[i16]@D	IR[i16]@D			IR[i16]@D				
NPC	PC4						PC4@D	PC4@D	PC4@D	
	I26						IR[i16]@D	IR[i26]@D	IR[i26]@D	
CMP	D1						RF.RD1			
	D2						RF.RD2			
E级	V1	RF.RD1	RF.RD1	RF.RD1	RF.RD1	RF.RD1				
	V2		RF.RD2	RF.RD2	RF.RD2					
	A1	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D				
	A2		IR[rt]@D	IR[rt]@D	IR[rt]@D					
	A3	IR[rt]@D		IR[rd]@D	IR[rd]@D	IR[rt]@D			31	
	E32	EXT	EXT							
	PC4								PC4@D	
ALU	A	V1@E	V1@E	V1@E	V1@E	V1@E				
	B	E32@E	E32@E	V2@E	V2@E	E32@E				
M级	V2		V2@E							
	A2		RD2@E							
	AO	ALU	ALU	ALU	ALU	ALU				
	A3	A3@E		A3@E	A3@E	A3@E				
	PC4								PC4@E	
DM	A	AO@M	AO@M	AO@M	AO@M	AO@M				
	WD		RD2@M							
W级	A3	A3@M		A3@M	A3@M	A3@M				
	PC4								PC4@M	
	AO	AO@M		AO@M	AO@M	AO@M				
	DR	DM								
RF	A3	A3@W		A3@W	A3@W	A3@W				
	WD	DR@W		AO@W	AO@W	AO@W			PC4@W	

综合无转发数据通路

方法

- ◆ 1) 合并同类项
- ◆ 2) 对于输入源在2个以上者，必须部署相应的MUX
 - MUX的输入信号：分别与各输入源相连接
 - MUX的输出信号：连接至功能部件的输入端

MPC控制信号
为PCSel

	lw	sw	add	sub	ori	beq	j	jal	jr	MUX	0	1	2
PC	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4 NPC	NPC	NPC	RF.RD1	MPC	ADD4	NPC	RF.RD1

注：本表仅截取了PC部分。

TIPS

MUX及其控制信号命名应符合某种规则。

MUX输入源的顺序与正确性无关，仅需在后续设计控制信号表达式确保一致性。

命名解读：MPC

M MUX

PC PC相关



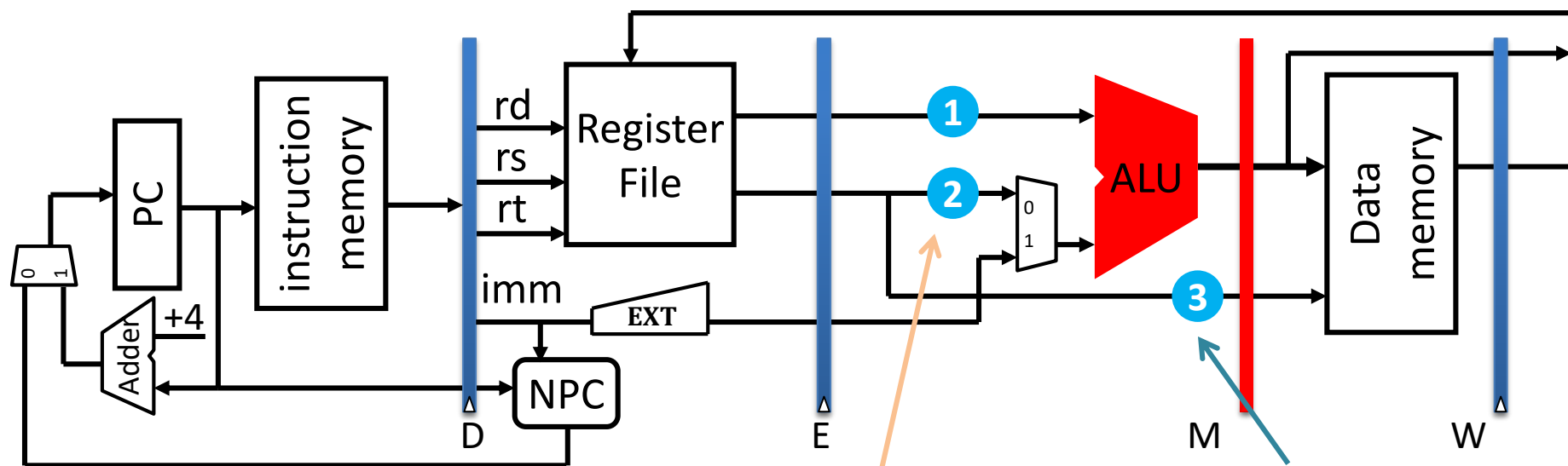
		lw	sw	add	sub	ori	beq	j	jal	jr	MUX	0	1	2
PC		ADD4	ADD4	ADD4	ADD4	ADD4	ADD4 NPC	NPC	NPC	RF.RD1	MPC	ADD4	NPC	RF.RD1
IM		PC	PC	PC	PC	PC	PC	PC	PC	PC	PC			
ADD4		PC	PC	PC	PC	PC	PC	PC	PC		PC			
D级	IR	IM	IM	IM	IM	IM	IM	IM	IM	IM	IM			
	NPC	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4		ADD4			
RF	A1	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D			IR[rs]@D	IR[rs]@D			
	A2		IR[rt]@D	IR[rt]@D	IR[rt]@D		IR[rt]@D				IR[rt]@D			
EXT		IR[i16]@D	IR[i16]@D			IR[i16]@D					IR[i16]@D			
NPC	PC4						PC4@D	PC4@D	PC4@D		PC4@D			
	I26						IR[i16]@D	IR[i26]@D	IR[i26]@D		IR[i26]@D			
CMP	D1						RF.RD1				RF.RD1			
	D2						RF.RD2				RF.RD2			
<div> 归并同类项并插入功能MUX， 形成完整的无转发数据通路。 与单周期、多周期数据通路综合方法完全相同 </div>											RF.RD1			
											RF.RD2			
											IR[rs]@D			
											IR[rt]@D			
											MA3E	IR[rt]@D	IR[rd]@D	
											EXT			
											PC4@D			
											V1@E			
											MALUB	V2@E	E32@E	
											V2@E			
W级											A2@E			
											ALU			
											A3@E			
											PC4@E			
DM	PC4								PC4@E		PC4@E			
	A	AO@M	AO@M	AO@M	AO@M	AO@M					AO@M			
W级	WD		RD2@M								V2@M			
	A3	A3@M		A3@M	A3@M	A3@M					A3@M			
	PC4								PC4@M		PC4@M			
	AO	AO@M		AO@M	AO@M	AO@M					AO@M			
RF	DR	DM									DM			
	A3	A3@W		A3@W	A3@W	A3@W					A3@W			
	WD	DR@W		AO@W	AO@W	AO@W			PC4@W		MRFWD	AO@W	DR@W	PC4@W

- 方法的流程概述
- 数据通路构造方法
 - ◆ 基础流水线规划
 - ◆ 建模指令RTL
 - ◆ RTL制导的独立数据通路
 - ◆ 综合无转发数据通路
 - ◆ 构造功能MUX控制表达式
 - ◆ 综合转发电路
- 暂停及转发的分析方法
- 暂停机制构造方法
- 转发机制构造方法
- 控制冒险处理机制

转发的设计思路

基本思路：

- ◆ 使用或传递rs或rt寄存器值功能部件和流水线寄存器，即为需求者
- ◆ 在需求者处增加转发MUX



注1：本图中流水线是不完整的
注2：本图仅给出了E和M之间的转发点

需求者（使用）

需求者（传递）

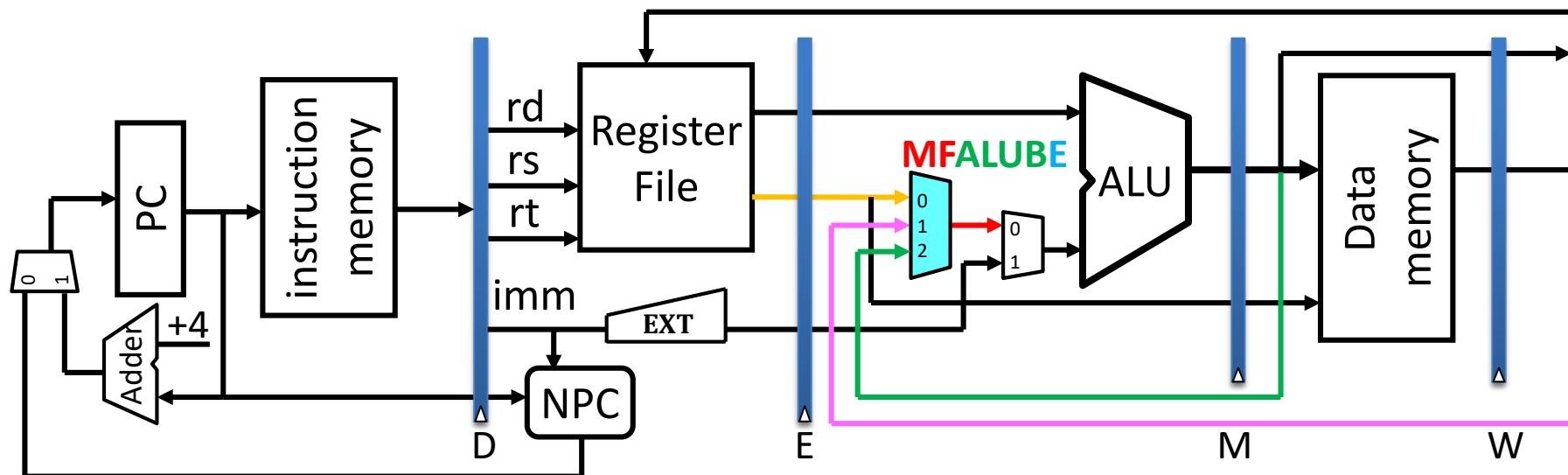


转发的设计思路

转发MUX

- ◆ 对于每个转发点，其后的所有各级流水线寄存器均需向其转发数据
- ◆ 转发MUX的路数等于 $N+1$
 - N ：后级转发数据的数量， 1 ：前级传递寄存器数据

【示例】 **MFALUBE**: ALU的B输入MUX前的转发MUX



MFALUBE

MF MUX Forward

ALUB ALU的B输入

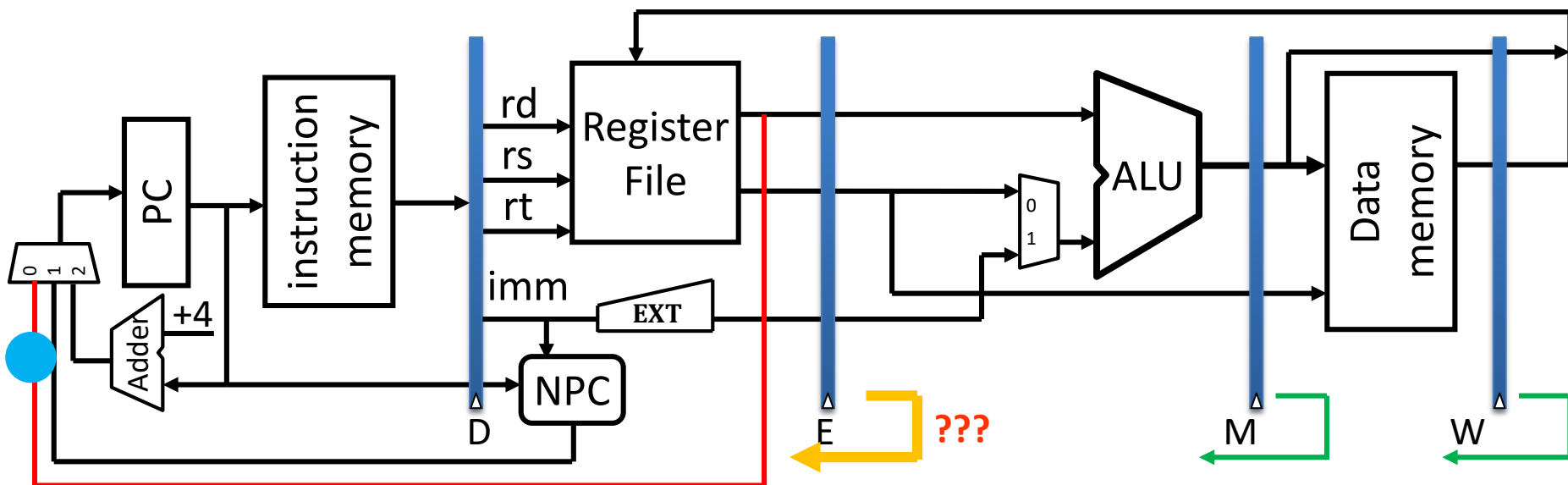
E E级

TIPS

- 1.建议规划转发MUX端口时，端口数字越大则其优先级越高
- 2.转发MUX命名应有规则和含义

转发的设计思路

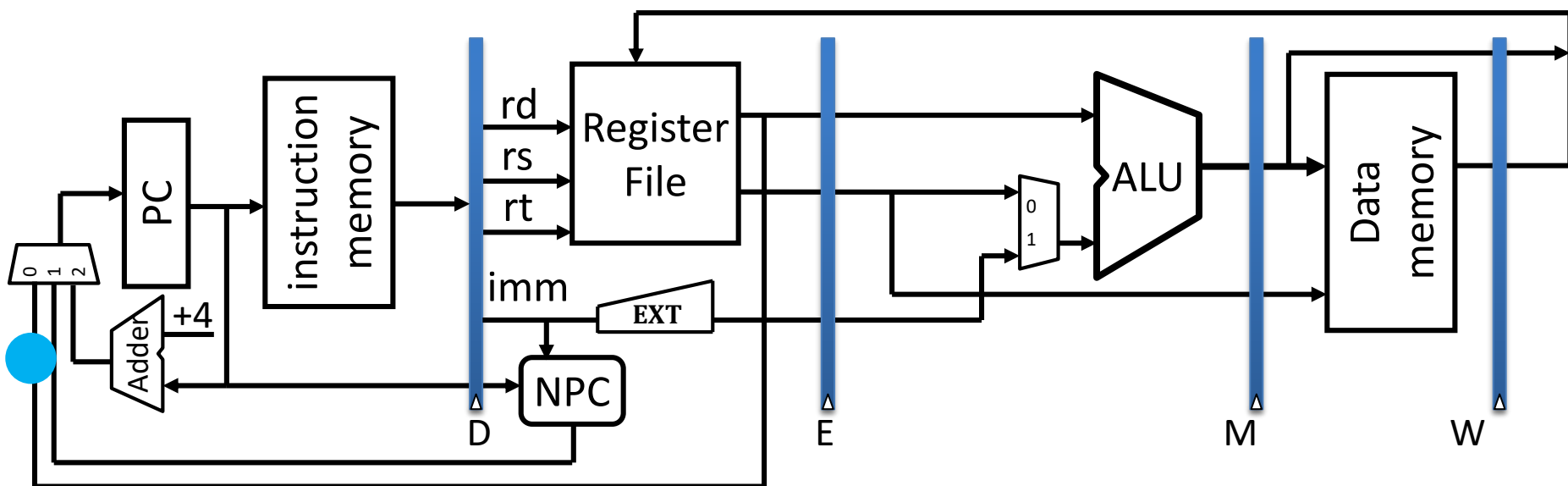
- 注意：D级寄存器也有转发点(并非只在E级之后)！
- 【示例jr】jr的RTL：PC \leftarrow R[rs]
 - 为了产生新PC，需要用到RF.RD1输出的V1值（rs寄存器值）
 - 因此NPC也存在一个转发点



Q: 考虑指令集{add, sub, ori, lw, sw, beq, j, jal, jr}。对于PC来说，转发数据除了M和W，是否应该包含E？

转发的设计思路

- Q: D级还包含2个转发点（图中并未给出相应的功能部件）。请问与哪条指令相关？
- A: beq!
 - beq比较电路前移至D级，因此rs和rt需要被从E、M、W转发



支持转发的完整数据通路

- 遍历数据通路的功能部件，找到所有与RF.RD1和RF.RD2相关的需求者

- 注意：V1--RF.RD1及V2--RF.RD2 的关联关系

- 示例：PC和ALU.B

- PC：输入2来自RF读出的第1个源操作数

- ALU.B：输入0来自V2，即RF读出的第2个源操作数

		lw	sw	add	sub	ori	beq	j	jal	jr	MUX	0	1	2
PC		ADD4	ADD4	ADD4	ADD4	ADD4	ADD4 NPC	NPC	NPC	RF.RD1	MPC	ADD4	NPC	RF.RD1
ALU	B	E32@E	E32@E	V2@E	V2@E	E32@E					MALUB	V2@E	E32@E	

该如何具体设计转发MUX呢？

MFALUBE

MFPCF



支持转发的完整数据通路

- 转发设计的基本思路：从需求者以后，所有可能存储新数据的流水寄存器，均需要向需求者转发数据。
- 示例：MALUB的输入0(来自V2@E)，需要替换为MFALUBE
 - 分析：ALU的后级为M和W。M存储ALU的计算结果，W存储ALU的计算结果以及DM的读出数据。
 - 结论：M和W均需要向MFALUB转发
 - M转发：AO；W转发：AO、DR

		lw	sw	add	sub	ori	beq	j	jal	jr	MUX	0	1	2
PC		ADD4	ADD4	ADD4	ADD4	ADD4	ADD4 NPC	NPC	NPC	RF.RD1	MPC	ADD4	NPC	RF.RD1
ALU	B	E32@E	E32@E	V2@E	V2@E	E32@E					MALUB	V2@E	E32@E	

TIPS

MUX的端口规划不影响设计，但最好按照数越大优先级越高，简单、易懂

	0	1	2	3
MFALUBE	V2@E	DR@W	AO@W	AO@M
优先级	低	中		高

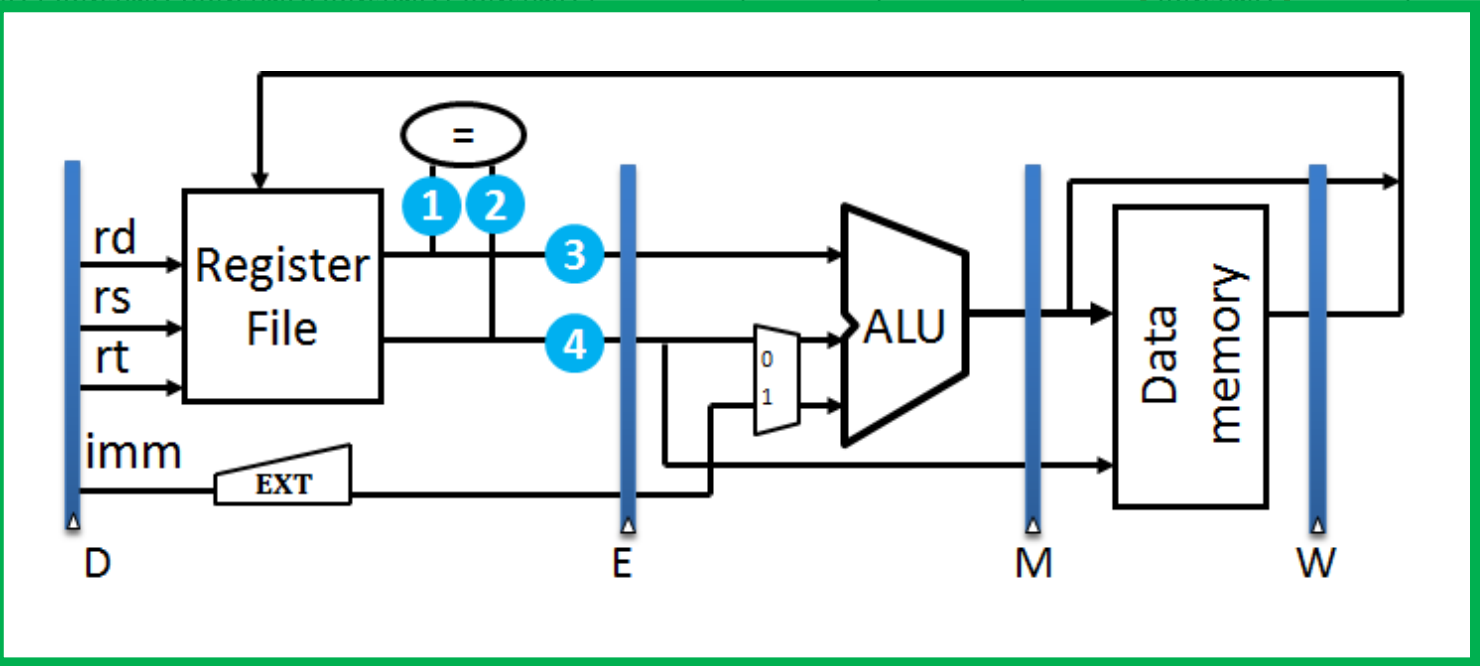
		lw	sw	add	sub	ori	beq	j	jal	jr	MUX	0	1	2
PC		ADD4	ADD4	ADD4	ADD4	ADD4	ADD4 NPC	NPC	NPC	RF.RD1	MPC	ADD4	NPC	RF.RD1
IM		PC	PC	PC	PC	PC	PC	PC	PC	PC	PC			
ADD4		PC	PC	PC	PC	PC	PC	PC	PC		PC			
D级	IR	IM	IM	IM	IM	IM	IM	IM	IM	IM	IM			
	NPC	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4		ADD4			
RF	A1	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D			IR[rs]@D	IR[rs]@D			
	A2		IR[rt]@D	IR[rt]@D	IR[rt]@D		IR[rt]@D				IR[rt]@D			
EXT		IR[i16]@D	IR[i16]@D			IR[i16]@D					IR[i16]@D			
NPC	PC4						PC4@D	PC4@D	PC4@D		PC4@D			
	I26						IR[i16]@D	IR[i26]@D	IR[i26]@D		IR[i26]@D			
CMP	D1						RF.RD1				RF.RD1			
	D2						RF.RD2				RF.RD2			
E级	V1	RF.RD1	RF.RD1	RF.RD1	RF.RD1	RF.RD1								
	V2		RF.RD2	RF.RD2	RF.RD2									
	A1	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D					IR[rs]@D			
	A2		IR[rt]@D	IR[rt]@D	IR[rt]@D						IR[rt]@D			
	A3	IR[rt]@D		IR[rd]@D	IR[rd]@D	IR[rt]@D			31		MA3E	IR[rt]@D	IR[rd]@D	
	E32	EXT	EXT								EXT			
	PC4								PC4@D		PC4@D			
ALU	A	V1@E	V1@E	V1@E	V1@E	V1@E					V1@E			
	B	E32@E	E32@E	V2@E	V2@E	E32@E					MALUB	V2@E	E32@E	
M级	V2		V2@E								V2@E			
	A2		RD2@E								A2@E			
	AO	ALU	ALU	ALU	ALU	ALU					ALU			
	A3	A3@E		A3@E	A3@E	A3@E					A3@E			
	PC4								PC4@E		PC4@E			
DM	A	AO@M	AO@M	AO@M	AO@M	AO@M					AO@M			
	WD		RD2@M								V2@M			
W级	A3	A3@M		A3@M	A3@M	A3@M					A3@M			
	PC4								PC4@M		PC4@M			
	AO	AO@M		AO@M	AO@M	AO@M					AO@M			
	DR	DM									DM			
RF	A3	A3@W		A3@W	A3@W	A3@W					A3@W			
	WD	DR@W		AO@W	AO@W	AO@W			PC4@W		MRFWD	AO@W	DR@W	PC4@W

		lw	sw	add	sub	ori	beq	j	jal	jr	MUX	0	1	2
PC		ADD4	ADD4	ADD4	ADD4	ADD4	ADD4 NPC	NPC	NPC	RF.RD1	MPC	ADD4	NPC	MFPCF
IM		PC	PC	PC	PC	PC	PC	PC	PC	PC	PC			
ADD4		PC	PC	PC	PC	PC	PC	PC	PC		PC			
D级	IR	IM	IM	IM	IM	IM	IM	IM	IM	IM	IM			
	NPC	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4		ADD4			
RF	A1	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D			IR[rs]@D	IR[rs]@D			
	A2		IR[rt]@D	IR[rt]@D	IR[rt]@D		IR[rt]@D				IR[rt]@D			
EXT		IR[i16]@D	IR[i16]@D			IR[i16]@D					IR[i16]@D			
NPC	PC4						PC4@D	PC4@D	PC4@D		PC4@D			
	I26						IR[i16]@D	IR[i26]@D	IR[i26]@D		IR[i26]@D			
CMP	D1						RF.RD1				MFCMP1D			
	D2						RF.RD2				MFCMP2D			
E级	V1	RF.RD1	RF.RD1	RF.RD1	RF.RD1	RF.RD1								
	V2		RF.RD2	RF.RD2	RF.RD2									
	A1	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D					IR[rs]@D			
	A2		IR[rt]@D	IR[rt]@D	IR[rt]@D						IR[rt]@D			
	A3	IR[rt]@D		IR[rd]@D	IR[rd]@D	IR[rt]@D			31		MA3E	IR[rt]@D	IR[rd]@D	
	E32	EXT	EXT								EXT			
	PC4								PC4@D		PC4@D			
ALU	A	V1@E	V1@E	V1@E	V1@E	V1@E					V1@E			
	B	E32@E	E32@E	V2@E	V2@E	E32@E					MALUB	MFALUBE	E32@E	
M级	V2		V2@E								MFV2M			
	A2		RD2@E								A2@E			
	AO	ALU	ALU	ALU	ALU	ALU					ALU			
	A3	A3@E		A3@E	A3@E	A3@E					A3@E			
	PC4								PC4@E		PC4@E			
DM	A	AO@M	AO@M	AO@M	AO@M	AO@M					AO@M			
	WD		RD2@M								MFWDM			
W级	A3	A3@M		A3@M	A3@M	A3@M					A3@M			
	PC4								PC4@M		PC4@M			
	AO	AO@M		AO@M	AO@M	AO@M					AO@M			
	DR	DM									DM			
RF	A3	A3@W		A3@W	A3@W	A3@W					A3@W			
	WD	DR@W		AO@W	AO@W	AO@W			PC4@W		MRFWD	AO@W	DR@W	PC4@W

		lw	sw	add	sub	ori	beq	j	jal	jr	MUX	0	1	2
	PC	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4 NPC	NPC	NPC	RF.RD1	MPC	ADD4	NPC	RF.RD1
	IM	PC	PC	PC	PC	PC	PC	PC	PC	PC	PC			
	ADD4	PC	PC	PC	PC	PC	PC	PC	PC		PC			
D级	IR	IM	IM	IM	IM	IM	IM	IM	IM	IM	IM			
	NPC	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4		ADD4			
RF	A1	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D			IR[rs]@D	IR[rs]@D			
	A2		IR[rt]@D	IR[rt]@D	IR[rt]@D	IR[rt]@D	IR[rt]@D				IR[rt]@D			
EXT		IR[i16]@D	IR[i16]@D								IR[i16]@D			
NPC	PC4								PC4@D		PC4@D			
	I26								I26]@D		IR[i26]@D			
CMP	D1										RF.RD1	1		
	D2										RF.RD2	2		
E级	V1	RF.RD1	RF.RD1								RF.RD1	3		
	V2		RF.RD2								RF.RD2	4		
	A1	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D					IR[rs]@D			
	A2													
	A3	IR[rt]@D												
	E32	EXT												
ALU	A	V1@D												
	B	E32@D												
M级	V2													
	A2													
	AO	ALU												
	A3	A3@D												
DM	PC4													
	A	AO@D												
W级	WD													
	A3	A3@D												
	PC4													
	AO	AO@D												
RF	DR	DM									DM			
	A3	A3@W		A3@W	A3@W	A3@W					A3@W			
	WD	DR@W		AO@W	AO@W	AO@W			PC4@W		MRFWD	AO@W	DR@W	PC4@W

TIPS

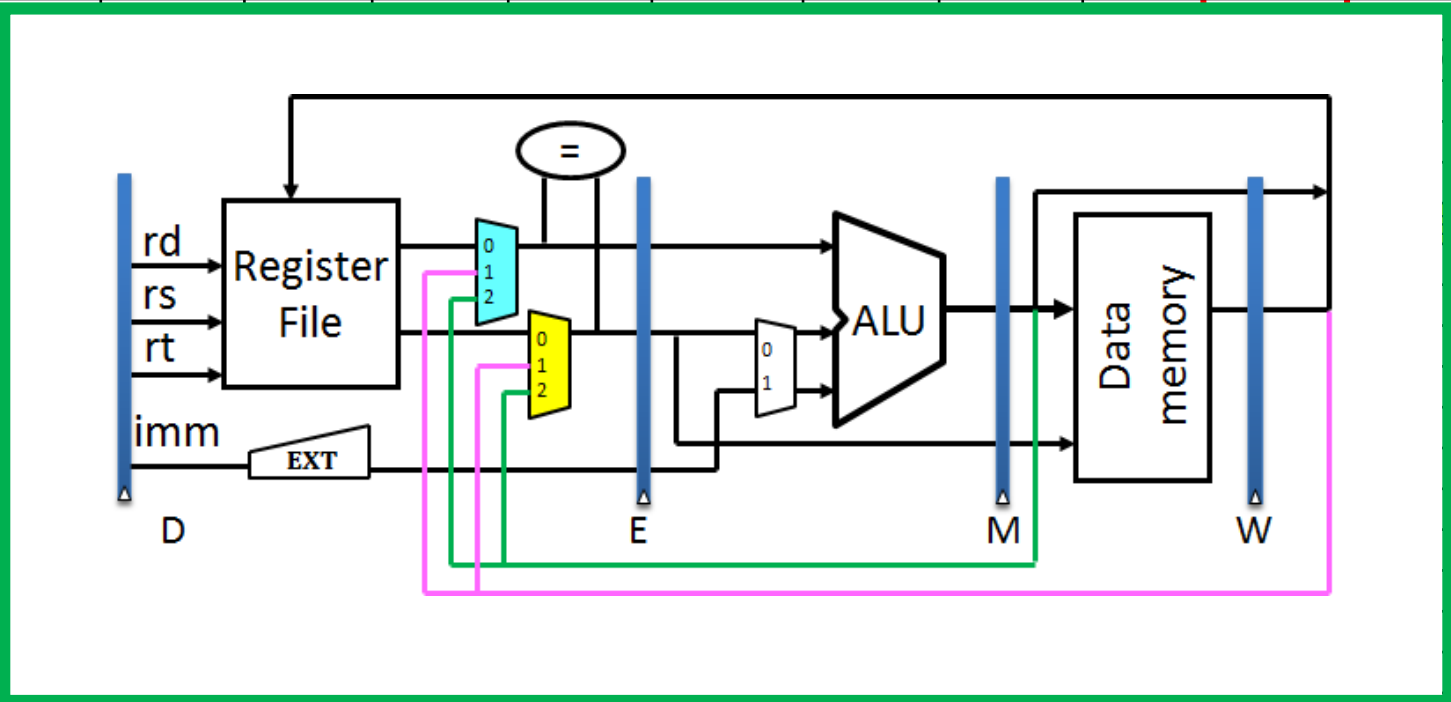
同级的同源需求者，只需设置一个转发MUX



		lw	sw	add	sub	ori	beq	j	jal	jr	MUX	0	1	2
	PC	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4 NPC	NPC	NPC	RF.RD1	MPC	ADD4	NPC	RF.RD1
	IM	PC	PC	PC	PC	PC	PC	PC	PC	PC	PC			
	ADD4	PC	PC	PC	PC	PC	PC	PC	PC		PC			
D级	IR	IM	IM	IM	IM	IM	IM	IM	IM	IM	IM			
	NPC	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4		ADD4			
RF	A1	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D			IR[rs]@D	IR[rs]@D			
	A2		IR[rt]@D	IR[rt]@D	IR[rt]@D	IR[rt]@D	IR[rt]@D				IR[rt]@D			
EXT		IR[i16]@D	IR[i16]@D								IR[i16]@D			
NPC	PC4								PC4@D		PC4@D			
	I26								I26]@D		IR[i26]@D			
CMP	D1										RF.RD1	1		
	D2										RF.RD2	2		
E级	V1	RF.RD1	RF.RD1								RF.RD1	3		
	V2		RF.RD2								RF.RD2	4		
	A1	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D					IR[rs]@D			
	A2													
	A3	IR												
	E32													
ALU	A													
	B													
M级	V2													
	A2													
	AO													
	A3													
DM	PC4													
	A													
W级	WD													
	A3													
	PC4													
	AO													
RF	A3	A3@W		A3@W	A3@W	A3@W					A3@W			
	WD	DR@W		AO@W	AO@W	AO@W			PC4@W		MRFWD	AO@W	DR@W	PC4@W

TIPS

同级的同源需求者，只需设置一个转发MUX



- 数据通路构造方法
 - ◆ 基础流水线规划
 - ◆ 建模指令RTL
 - ◆ RTL制导的独立数据通路
 - ◆ 综合无转发数据通路
 - ◆ 综合转发电路
 - ◆ 构造功能MUX控制表达式
- 暂停及转发的分析方法
- 暂停机制构造方法
- 转发机制构造方法
- 控制冒险处理机制

构造功能MUX控制信号表达式

□ 数据通路中包含2类MUX：功能MUX、转发MUX

- ◆ 功能MUX：与指令执行的功能（即操作语义）相关
- ◆ 转发MUX：与指令执行的性能相关

		lw	sw	add	sub	ori	beq	j	jal	jr	MUX	0	1	2
PC		ADD4	ADD4	ADD4	ADD4	ADD4	ADD4 NPC	NPC	NPC	RF.RD1	MPC	ADD4	NPC	MFPC
CMP	D1						RF.RD1				MFCMP1D			
	D2						RF.RD2				MFCMP2D			
E级	A3	IR[rt]@D		IR[rd]@D	IR[rd]@D	IR[rt]@D			31		MA3E	IR[rt]@D	IR[rd]@D	
ALU	B	E32@E	E32@E	V2@E	V2@E	E32@E					MALUB	MFALUB	E32@E	
M级	V2		V2@E								MFV2M			
DM	WD		RD2@M								MFWD			
RF	WD	DR@W		AO@W	AO@W	AO@W			PC4@W		MRWD	AO@W	DR@W	PC4@W

□ 目前先构造功能MUX控制信号表达式

- ◆ 在暂停与转发控制部分讲解转发MUX控制信号表达式



构造功能MUX控制信号表达式

- 示例：构造PC的功能MUX的控制信号PCSel表达式

	lw	sw	add	sub	ori	beq	j	jal	jr	MUX	0	1	2
PC	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4 NPC	NPC	NPC	RF.RD1	MPC	ADD4	NPC	MFPCF

$$PCSel = (lw + sw + add + sub + ori + beq \& !Zero) ? \text{ADD4} :$$

$$(beq \& Zero + j + jal) ? NPC : MFPCF$$

- 数据通路表：决定了MUX控制信号的取值

- 注意1：表格和表达式的颜色对应关系
- 注意2：转发MUX与寄存器值之间的关系
 - 示例：MFPCF的某路输入就是RF.RD1

```

`define ADD4 2'b00
`define NPC  2'b01
...
    
```

TIPS

通过宏提高代码可读性、可维护性



- 数据通路构造方法
 - ◆ 基础流水线规划
 - ◆ 建模指令RTL
 - ◆ RTL制导的独立数据通路
 - ◆ 综合无转发数据通路
 - ◆ 综合转发电路
 - ◆ 构造功能MUX控制表达式
- 暂停及转发的分析方法
- 暂停机制构造方法
- 转发机制构造方法
- 控制冒险处理机制

流水线的执行特点及数据冒险对策的基本构思

- 目前的流水线的基本特点是：按序发射，按序完成
 - ◆ 一旦当前指令阻塞，后续指令就被阻塞
- 这意味着，对于需要暂停的指令来说，无论在哪级暂停，其后续指令都不能执行
- 既然如此，那么对于数据相关的暂停和转发，就可以分成两个独立环节
 - ◆ 在D级检测

流水线的执行特点及数据冒险对策的基本构思

- 在D级就暂停住指令可能是不合理的。
- 下面的例子，由于sub被冻结在D直至冒险解除，因此beq指令在cycle4才进入D

		PC+4		RF(读)		ALU		DM	
地址	指令	CLK	PC	IM	D	E	M	W	RF
0	lw \$t0, 0(\$t1)	↑ 1	0	lw	lw				
			4	sub	写t0 3				
4	sub \$t3, \$t0, \$t2	↑ 2	4	sub	sub	lw			
			8	and	读t0 1	写t0 2			
8	beq	↑ 3	8	beq	sub		lw		
			8	beq	读t0 1	nop	写t0 1		
12	XXX	↑ 4	8	beq	beq	sub		lw	
			12	XXX	读t0 1	读t0 0	nop	新t0 0	
16									

流水线的执行特点及数据冒险对策的基本构思

- 如果允许sub前进直至必须暂停，那么beq在cycle3就进入了D
- 由于beq是在D级执行的，因此sub进入E级处于执行时，D级的beq也能同时执行
- 对于5级流水线来说，只有b和j在D级执行，因此是受益指令

		PC+4		RF(读)		ALU		DM	
地址	指令	CLK	PC	IM	D	E	M	W	RF
0	lw \$t0, 0(\$t1)	↑ 1	0	lw	lw				
			4	sub	写t0 3				
4	sub \$t3, \$t0, \$t2	↑ 2	4	sub	sub	lw			
			8	beq	读t0 1	写t0 2			
8	beq	↑ 3	8	beq	beq		lw		
			12	xxx		sub	写t0 1		
12	xxx	↑ 4	12	xxx	xxx	sub		lw	
			16	yyy		读t0 0	nop	新t0 0	
16	yyy								

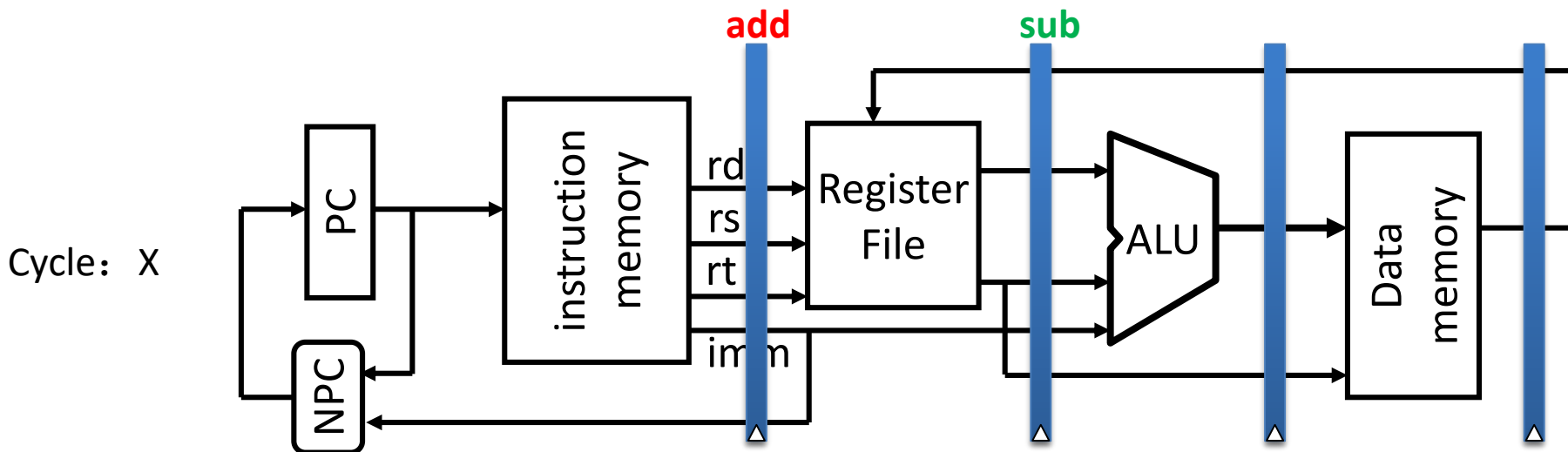
流水线的执行特点及数据冒险对策的基本构思

- 如果采用该方案，则就必须再增加标记：
 - ◆ 标记每个流水段是否执行结束
 - ◆ 当然，对于5级流水线来说，这个标记可以只在D级就可以了。
 - 虽然E和M都是执行（分别对应ALU和DM），但M级只有转发了
- 当指令是b/j时，标志置位，意味着可以再继续执行，否则就必须阻塞了
- 该方案将使得流水线的控制更加复杂
- 我认为由于受益指令非常有限，而且方法学重点在于解决100%的覆盖性，因此仍然保持原设计：D级阻塞需要暂停的指令

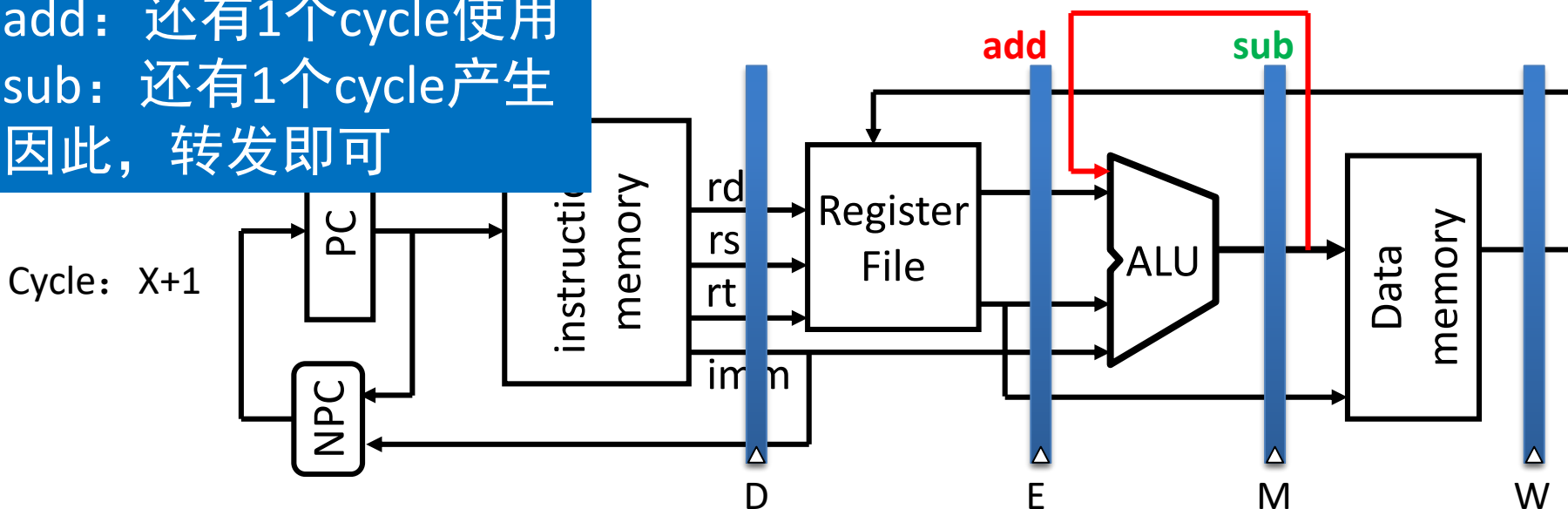
暂停or转发

假设add与sub的rs相关

- 案例1：暂停add？还是可以转发sub的结果？



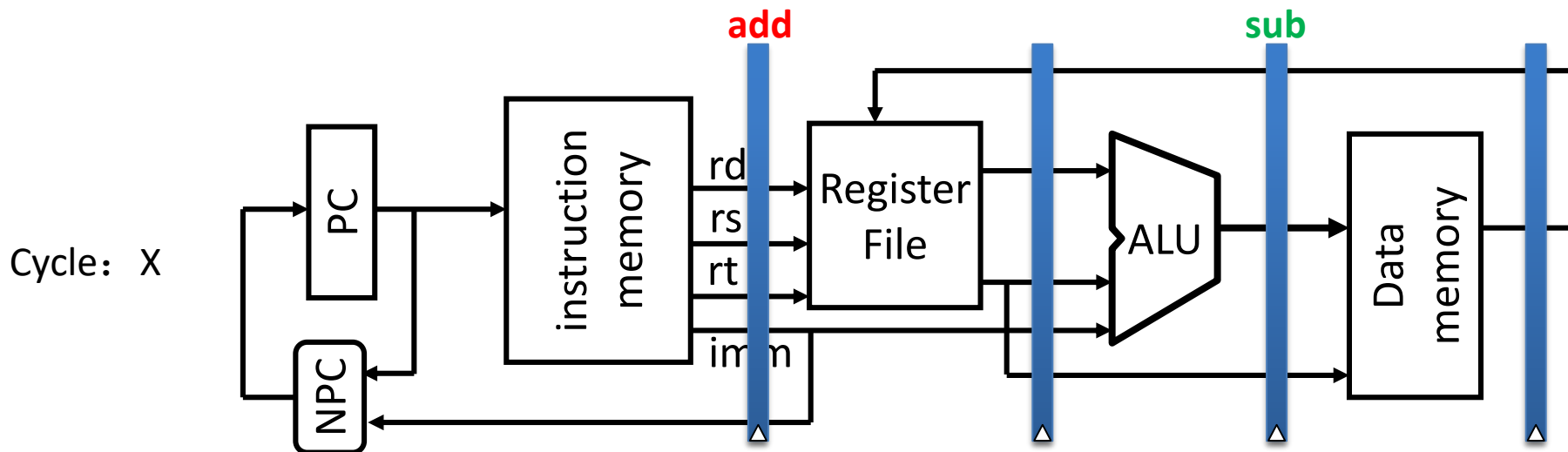
add: 还有1个cycle使用
sub: 还有1个cycle产生
因此，转发即可



暂停or转发

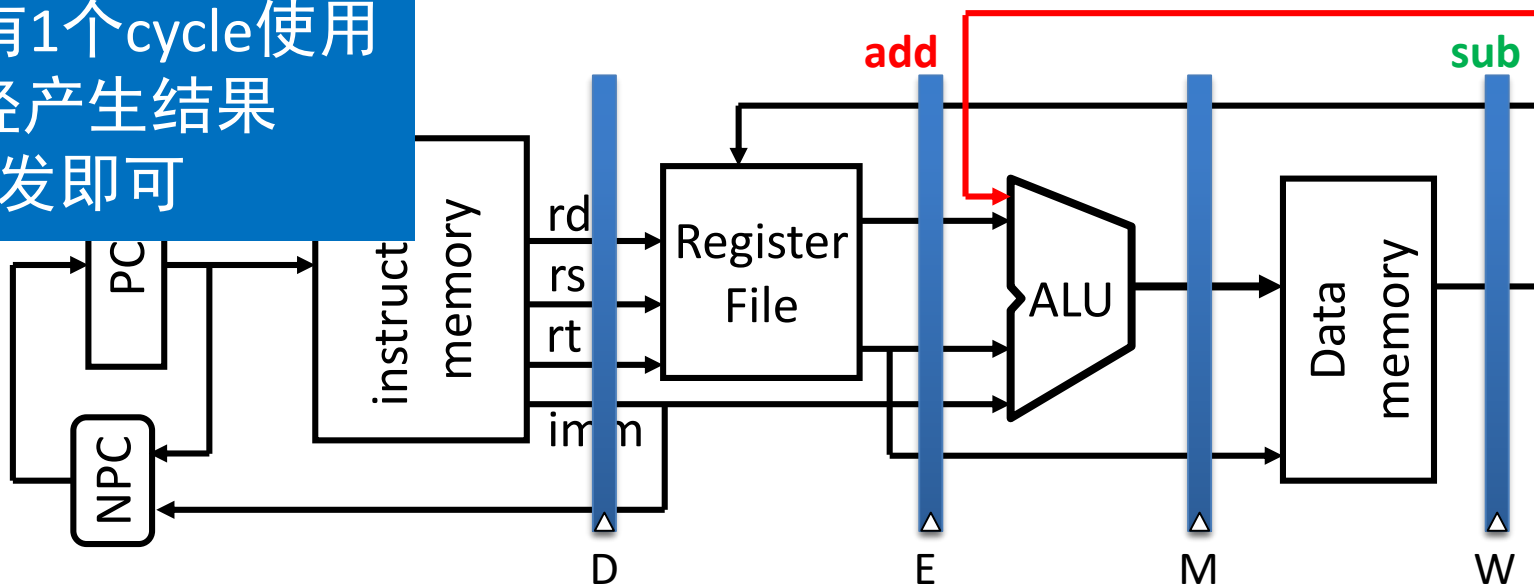
假设add与sub的rs相关

- 案例1：暂停add？还是可以转发sub的结果？



add: 还有1个cycle使用
sub: 已经产生结果
因此，转发即可

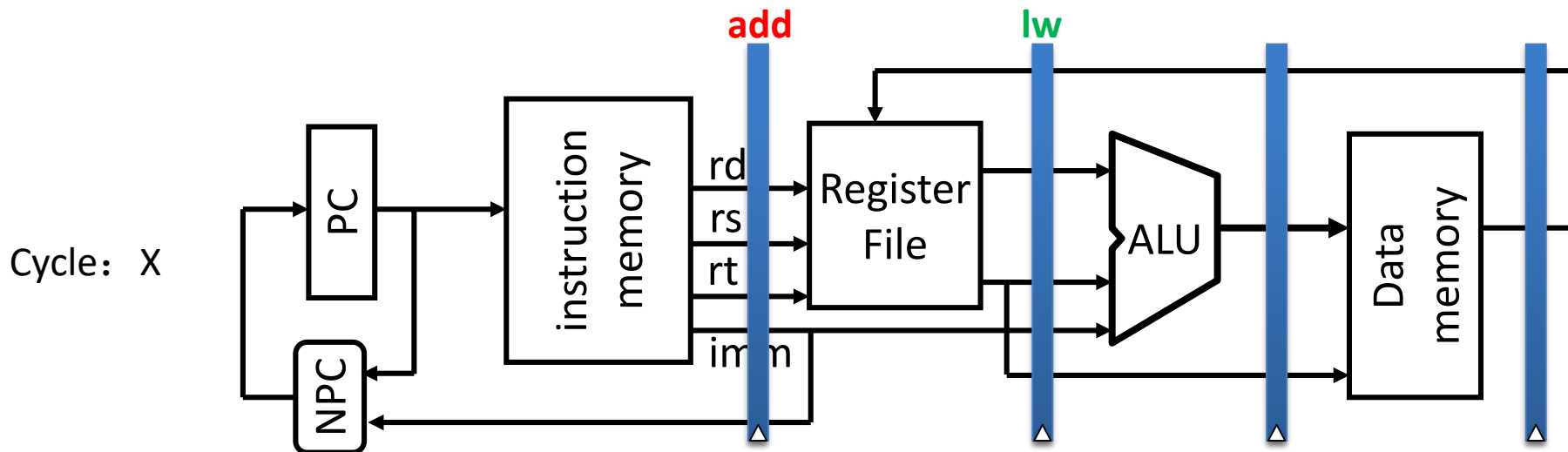
Cycle: X+1



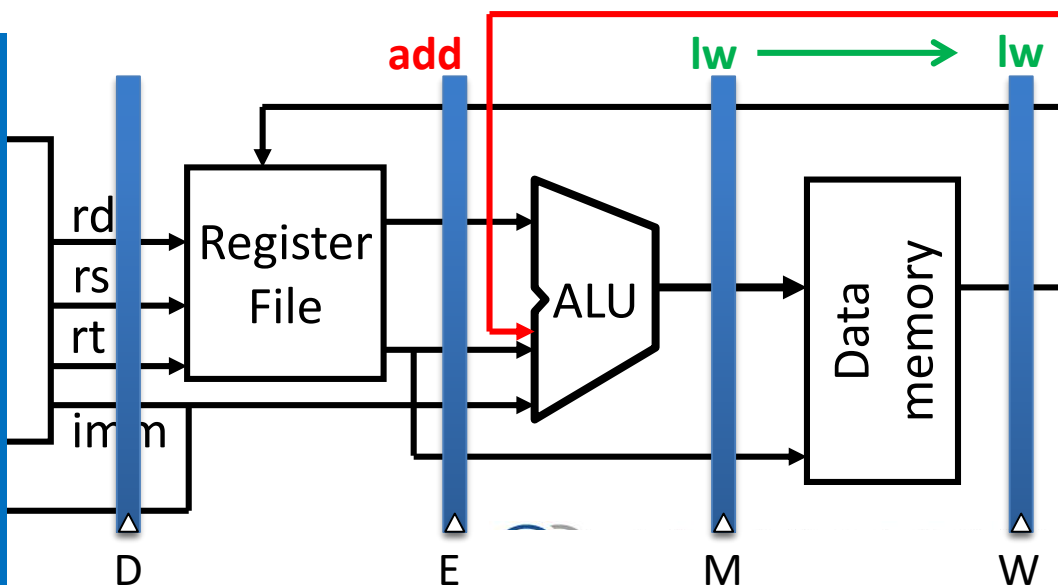
暂停or转发

假设add与lw的rt相关

- 案例3：暂停add？还是可以转发sub的结果？



add: 还有1个cycle使用
lw: 还有2个cycle产生
数据产生时间比使用时
间滞后1个cycle。
因此，最佳方案是：
1) add暂停1个cycle
2) 然后转发数据



数据冒险：需求与供给能否匹配？

- 需求者：使用或传递寄存器值的**功能部件**和**流水线寄存器**
 - ◆ 例1：add/sub/or的需求在E级的ALU
 - ◆ 例2：j指令不需要读取任何寄存器，因此j指令没有需求
- 供给者：保存有reg新结果的**流水线寄存器**
 - ◆ 例1：所有运算类指令的供给者是M级和W级
 - ◆ 例2：load类指令的供给者是W级
- 数据冒险可以转化为：需求与供给的匹配
 - ◆ 暂停：结果产生的时间**晚于**指令到达需求者时（前）的时间
 - ◆ 转发：结果产生的时间**早于/等于**指令到达需求者时（前）的时间
- Q：如果有多个供给者，M级、W级哪个值是最新值？

需求者的最晚时间模型

- T_{use} (time-to-use): 指令进入D级后, 其后的某个功能部件再经过多少cycle就**必须**要使用寄存器值
- 特点1: T_{use} 是静态值, 读取操作数的时间上限
 - ◆ 例如, R型计算类指令的 T_{use} 为1 (rs/rt均在E级使用)
- 特点2: 同一条指令可以有2个不同的 T_{use}
 - ◆ 例1, store型指令的 T_{use} 分别为1 (rs在E级使用) 和2 (rt在M级使用)
 - ◆ 例2: 假设存在一条指令要读取3个寄存器, 那么就可能有3个不同的 T_{use}
 - 【注】要支持读取3个寄存器, 则必须修改RF的设计。MIPS并无这种需求。
- Beq指令: 由于寄存器比较功能被前移至D级, 因此 $T_{use}=0$
 - ◆ 如果beq不前移, 则 T_{use} 必然不为0

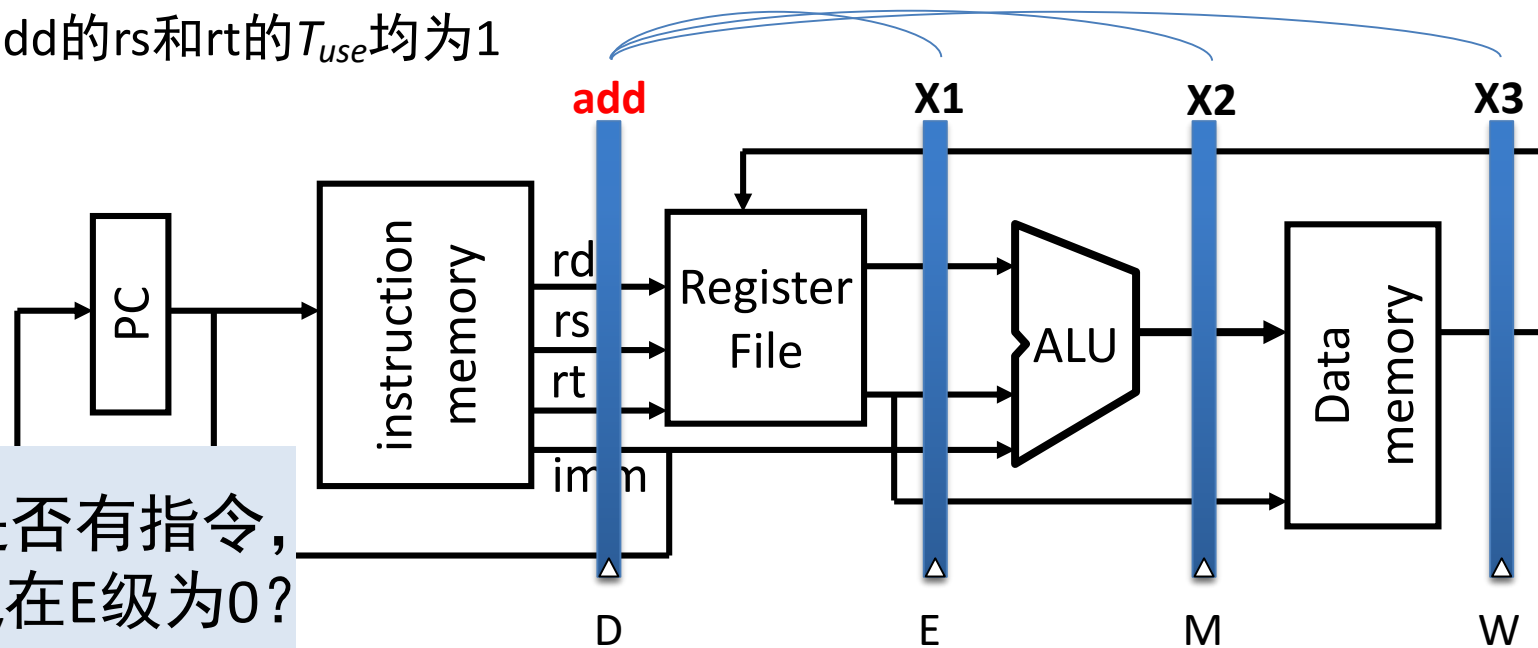
供给者的最早时间模型

- T_{new} (time-to-new)：位于E级及其后各级的指令，再经过多少周期能够产生要写入寄存器的结果
- 特点1：动态值，随着指令的流动，该值在不断减小，直至0
- 特点2：一条指令可以有多个不同的 T_{new}
- 例如，R型计算类指令的 T_{new} 为1或0
 - ◆ 1：指令位于E级，ALU正在计算
 - ◆ 0：指令位于M级或W级，结果已经存储在相应级
- 例如，load型计算类指令的 T_{new} 为2，1，0
 - ◆ 2：指令位于E级，尚未读取存储器
 - ◆ 1：指令位于M级，正在读取存储器
 - ◆ 0：指令位于W级，结果已经存储在W级

暂停转发的基本分析方法

- 以add的rs为例：将其 T_{use} 与位于E/M/W各级指令的 T_{new} 比较。如果 T_{new} 大，则只能暂停，否则转发

- add的rs和rt的 T_{use} 均为1



add的rs暂停转发分析矩阵

F 转发
S 暂停

$T_{new} \backslash T_{use}^{rs}$	E级			M级		W级
	0	1	2	0	1	0
1	F	F	S	F	F	F

← 各级可能的取值范围

- 数据通路构造方法
 - ◆ 基础流水线规划
 - ◆ 建模指令RTL
 - ◆ RTL制导的独立数据通路
 - ◆ 综合无转发数据通路
 - ◆ 综合转发电路
 - ◆ 构造功能MUX控制表达式
- 暂停及转发的分析方法
- 暂停机制构造方法
- 转发机制构造方法
- 控制冒险处理机制

构造指令集的 T_{use}

□ 思路：结合流水线架构，逐条指令构造 T_{use} 及 T_{new}

□ T_{use} 注意事项

- ◆ 1) 只关注每条指令的操作语义
- ◆ 2) 指令可能有2个不同的 T_{use} ，如sw
- ◆ 3) 指令集或流水线架构的变化，均可能导致 T_{use} 变化
 - 例如：流水线从5级变为6级且第5级为访存，则sw的rt将会延后1级被使用，故rt的 T_{use} 会变为{0,1,2,3}

	T_{use}	
	rs	rt
add	1	1
sub	1	1
andi	1	
ori	1	
lw	1	
sw	1	2
beq	0	0
jr	0	
	{0,1}	{0,1,2}

构造指令集的 T_{new}

思路：结合流水线架构，逐条指令构造 T_{use} 及 T_{new}

T_{new} 注意事项：

一旦减为0，则不再继续减少！

- 0：有效结果已经产生了
- 非0：有效结果尚未产生

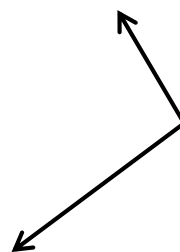
为了便于分析，用产生结果的功能部件来代表指令

例如，ALU可以代表所有的计算类指令

E			M			W		
ALU	DM	PC	ALU	DM	PC	ALU	DM	PC
1	2	0	0	1	0	0	0	0

指令	功能部件	T_{new}		
		E	M	W
add	ALU	1	0	0
sub	ALU	1	0	0
andi	ALU	1	0	0
ori	ALU	1	0	0
lw	DM	2	1	0
sw				
beq				
jal	PC	0	0	0

产生结果的功能部件



根据 T_{use} 和 T_{new} 构造策略矩阵

结合指令 T_{use}/T_{new} ，分别构造rs/rt寄存器的策略矩阵

◆ $T_{new} > T_{use}$ ：只能暂停

- 结果产生的时间太晚，不可能通过转发实现，必须暂停

◆ $T_{new} \leq T_{use}$ ：通过转发可以解决冲突

rs策略矩阵

$T_{new} \backslash T_{use}$	E			M			W		
	ALU	DM	PC	ALU	DM	PC	ALU	DM	PC
	1	2	0	0	1	0	0	0	0
0	S	S	F	F	S	F	F	F	F
1	F	S	F	F	F	F	F	F	F

rt策略矩阵

$T_{new} \backslash T_{use}$	E			M			W		
	ALU	DM	PC	ALU	DM	PC	ALU	DM	PC
	1	2	0	0	1	0	0	0	0
0	S	S	F	F	S	F	F	F	F
1	F	S	F	F	F	F	F	F	F
2	F	F	F	F	F	F	F	F	F

根据策略矩阵构造暂停条件的一般性方法

rs策略矩阵

$T_{\text{new}} \backslash T_{\text{use}}$	E			M			W		
	ALU	DM	PC	ALU	DM	PC	ALU	DM	PC
	1	2	0	0	1	0	0	0	0
0	S	S	F	F	S	F	F	F	F
1	F	S	F	F	F	F	F	F	F

rt策略矩阵

$T_{\text{new}} \backslash T_{\text{use}}$	E			M			W		
	ALU	DM	PC	ALU	DM	PC	ALU	DM	PC
	1	2	0	0	1	0	0	0	0
0	S	S	F	F	S	F	F	F	F
1	F	S	F	F	F	F	F	F	F
2	F	F	F	F	F	F	F	F	F

$\text{Stall_RS0_e1} = \dots$
 $\text{Stall_RS0_e2} = \dots$
 $\text{Stall_RS0_m1} = \dots$
 $\text{Stall_RS1_e2} = \dots$

$\text{Stall_RS} = \text{Stall_RS0_e1} \mid$
 $\text{Stall_RS0_e2} \mid$
 \dots

+

$\text{Stall_RT0_e1} = \dots$
 $\text{Stall_RT0_e2} = \dots$
 $\text{Stall_RT0_m1} = \dots$
 $\text{Stall_RT1_e2} = \dots$

$\text{Stall_RT} = \text{Stall_RT0_e1} \mid$
 $\text{Stall_RT0_e2} \mid$
 \dots

↓

$\text{Stall} = \text{Stall_RS} \mid$
 Stall_RT

Q:
如何表示 T_{new} 与 T_{use} ?

如何用变量表示 T_{use} 与 T_{new} ?

□ T_{use} :

- ◆ 对于rs和rt, 可以分别各用一组译码变量与取值相对应

□ T_{new} :

- ◆ 每条指令有多个, 因此需要在E/M/W都要有**相关信息**
- ◆ 需要一种相对简洁的表示方法

指令	T_{use}	
	rs	rt
add	1	1
sub	1	1
andi	1	
ori	1	
lw	1	
sw	1	2
beq	0	0
jr	0	
	{0,1}	{0,1,2}

指令	功能部件	T_{new}		
		E	M	W
add	ALU	1	0	0
sub	ALU	1	0	0
andi	ALU	1	0	0
ori	ALU	1	0	0
lw	DM	2	1	0
sw				
beq				
jal	PC	0	0	0

用变量表示 T_{use}

□ 用变量来表示策略矩阵的 T_{use}

- ◆ rs有2个取值，因此对应2个变量
- ◆ rt有3个取值，因此对应3个变量

$T_{use_RS0} = beq + jr$

$T_{use_RS1} = add + sub + andi + \dots + sw$

$T_{use_RT0} = \dots$

$T_{use_RT1} = \dots$

$T_{use_RT2} = sw$

	Tuse	
	rs	rt
add	1	1
sub	1	1
andi	1	
ori	1	
lw	1	
sw	1	2
beq	0	0
jr	0	
	{0,1}	{0,1,2}



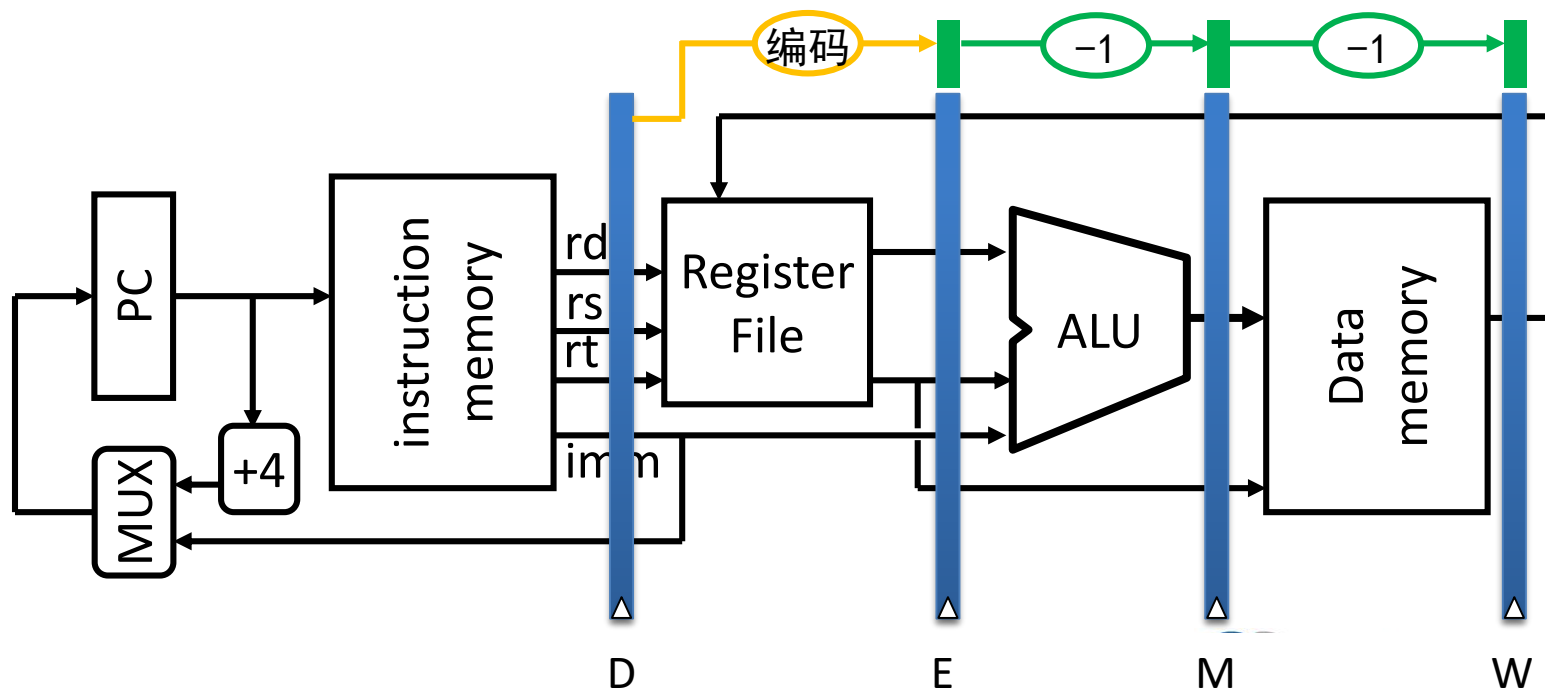
建模 T_{new} 计数器

在E/M/W分别设置1个2位的计数器

- 在D级，根据指令产生编码值
- 逐级减1，直至0

指令类别	T_{new} 初值
运算类	1
读存储类	2
函数调用类	0

注意：还需要增加1个主控制器产生的写使能才能完整、准确的表达：**这是一条写RF的指令&结果是否有效**



建模 T_{new} 计数器

- 指令集 T_{new} 矩阵的E级就是流水线中 T_{new} 的初值

Q
为什么没有定义
 T_{new_W} ?

指令	功能 部件	T_{new}		
		E	M	W
add	ALU	1	0	0
sub	ALU	1	0	0
andi	ALU	1	0	0
ori	ALU	1	0	0
lw	DM	2	1	0
sw				
beq				
jal	PC	0	0	0

指令类别	T_{new} 初值
运算类	1
读存储类	2
函数调用类	0

```

`define T_ALU    2'b01
`define T_DM     2'b10
`define T_PC     2'b00
    
```

```

reg [1:0] Tnew_E, Tnew_M ;

always @(.....)
    if (add | sub | andi | ori)
        Tnew_E <= `T_ALU ;
    else if (lw)
        Tnew_E <= `T_DM ;
    .....
    
```

构造暂停条件表达式

分别构造rs和rt的暂停条件

示例：rs的暂停条件

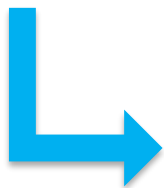
$T_{new} \backslash T_{use}^{rs}$	E			M			W		
	ALU	DM	PC	ALU	DM	PC	ALU	DM	PC
	1	2	0	0	1	0	0	0	0
0	S	S	F	F	S	F	F	F	F
1	F	S	F	F	F	F	F	F	F



```
`define A1 IR[25:21]
```

```

Stall_RS0_E1 = Tuse_RS0 & (Tnew_E == 2'b01) & (`A1 == A3_E) & W_E
Stall_RS0_E2 = Tuse_RS0 & (Tnew_E == 2'b10) & (`A1 == A3_E) & W_E
Stall_RS0_M1 = Tuse_RS0 & (Tnew_M == 2'b01) & (`A1 == A3_M) & W_M
Stall_RS1_E2 = Tuse_RS1 & (Tnew_E == 2'b10) & (`A1 == A3_M) & W_E
    
```



```

Stall_RS = Stall_RS0_E1 |
           Stall_RS0_E2 |
           Stall_RS1_E2 |
           Stall_RS0_M1
    
```

构造暂停条件表达式

- 将rs和rt的暂停“或”起来，就形成了总的暂停条件

$$\text{Stall} = \text{Stall_RS} \mid \text{Stall_RT}$$



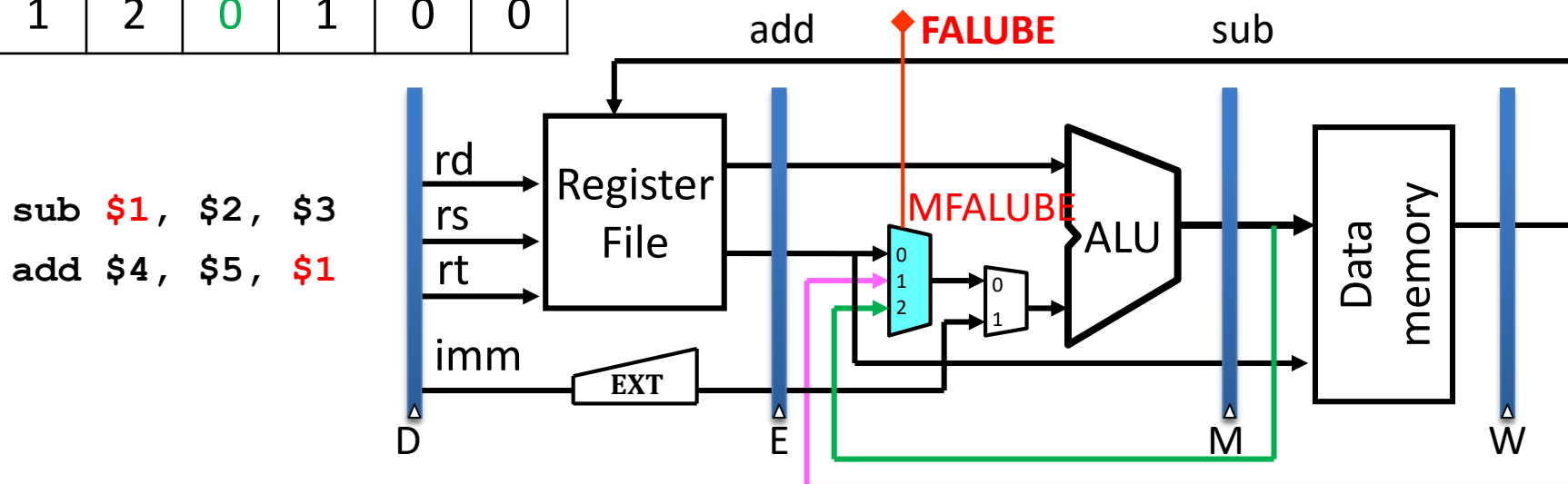
- 数据通路构造方法
 - ◆ 基础流水线规划
 - ◆ 建模指令RTL
 - ◆ RTL制导的独立数据通路
 - ◆ 综合无转发数据通路
 - ◆ 综合转发电路
 - ◆ 构造功能MUX控制表达式
- 暂停及转发的分析方法
- 暂停机制构造方法
- 转发机制构造方法
- 控制冒险处理机制

转发控制的基本分析方法

- 转发的核心：控制转发MUX选择最新的数据
- 【案例分析】ALU的B输入转发MUX转发M级数据的条件：
 - ◆ 1) 读寄存器编号与写寄存器编号相同：A2_E == A3_M
 - ◆ 2) M级存储了有效结果（用T_{new}与写使能这2个信息表示结果是否有效）

E		M		W	
ALU	DM	ALU	DM	ALU	DM
1	2	0	1	0	0

Q: 我们是否需要增加E级必须是add (即必须是读rt) 这个条件?



转发控制的基本分析方法

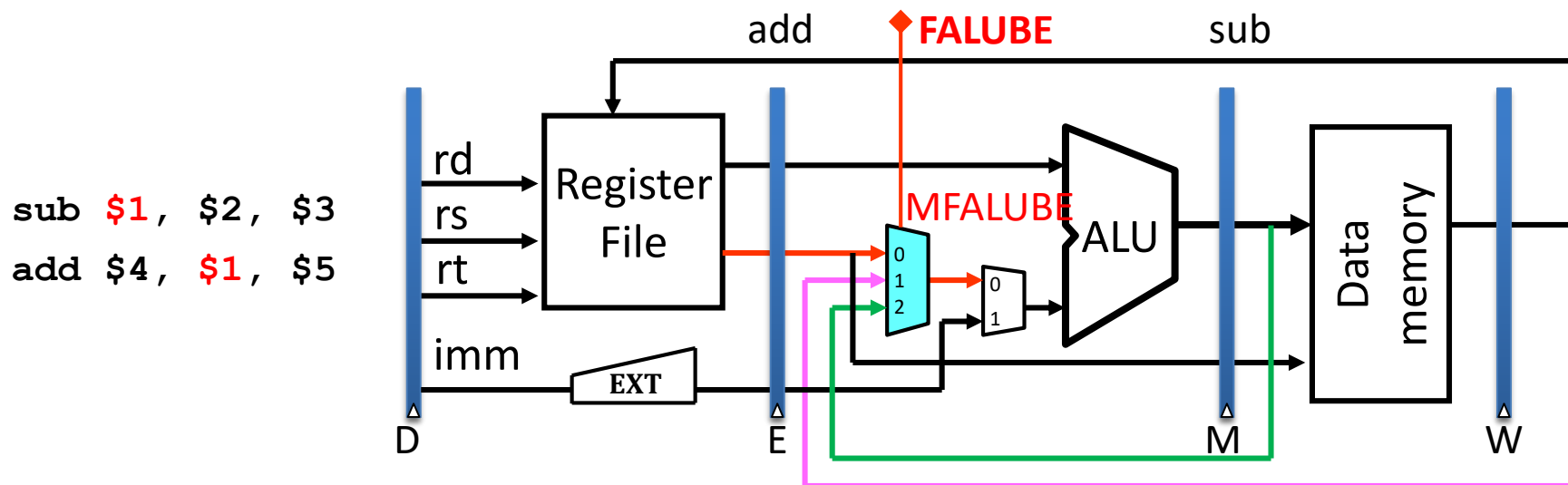
分析：E级指令存在如下3种可能

- ◆ 1) 无效的rt域：如j
 - ◆ 2) 有效的rt域：写rt，如ori/lw
 - ◆ 3) 有效的rt域：**读rt**，如add/sub
- ◆ 1和2，转发了也不会导致错误；3，转发为必须

1和2：

E级指令在E级执行正确性与是否读rt寄存器无关；由此可知转发不影响E级指令的正确性。

结论：故无需进一步增加条件

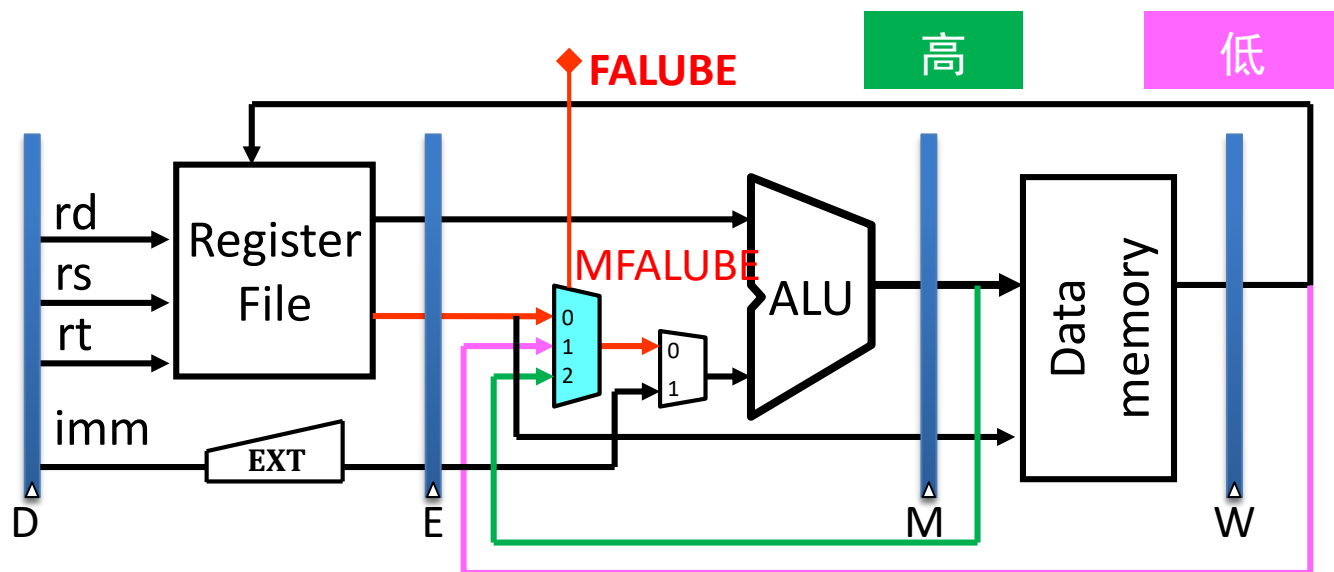


转发控制的基本分析方法

- ❑ 优先级问题：当流水线中2条以上指令都写同一个寄存器，意味着2级以上流水寄存器均包含新值，则转发MUX选择哪级流水寄存器的新值呢？
- ❑ A：新值距离D越近，则意味着新值越“新鲜”。因此距离D越近，转发的优先级越高；反之则越低。
- ❑ 【示例MFRTE】E级转发优先级最高；W级次之；RF的输出最低

TIPS

建议规划转发MUX端口时，端口数字越大则其优先级越高



转发控制的基本分析方法

- 转发MUX控制信号表达式（注意优先级）
- 示例：ALU的B端的转发MUX控制信号~FALUBE

```

`define M2E_ALU 3 //M向E转发ALU结果
`define W2E_ALU 2 //W向E转发ALU结果
`define W2E_DM 1 //W向E转发DM结果

```

TIPS

书写表达式时，尽量多使用宏，增加可读性。

FALUBE =

```

((A2_E==A3_M) & (Tnew_M==2'b00) & W_M ? `M2E_ALU :
((A2_E==A3_W) & W_W ? `W2E_ALU :
((A2_E==A3_W) & W_W ? `W2E_DM :
0

```

高
↓
低

E		M		W	
ALU	DM	ALU	DM	ALU	DM
1	2	0	1	0	0

	0	1	2	3
MFALUBE	V2@E	DR@W	AO@W	AO@M

指令集的T_{new}
注意：仅为示意，可能不完整

转发MUX
注：可能不完整

- 数据通路构造方法
 - ◆ 基础流水线规划
 - ◆ 建模指令RTL
 - ◆ RTL制导的独立数据通路
 - ◆ 综合无转发数据通路
 - ◆ 构造功能MUX控制表达式
 - ◆ 综合转发电路
- 暂停及转发的分析方法
- 暂停机制构造方法
- 转发机制构造方法
- 控制冒险处理机制

控制冒险处理机制

- ❑ 分歧点1: 是否实现延迟槽
 - ◆ 如果实现, 需要注意jal及jalr指令应保存PC+8(需要在D级再部署一个PC+4)
- ❑ 分歧点2: 比较功能是否前移至ID阶段
- ❑ 课程要求: 实现延迟槽, 并且比较前移至ID阶段

延迟槽 前移	是	否
	是	B类: 有条件清除IF/ID J类: 无条件清除IF/ID
否	硬件无需处理 编译调度指令	B类: 有条件清除IF/ID、ID/EX J类: 无条件清除IF/ID、ID/EX、EX/MEM

Q: JAL、JALR的回写寄存器怎么处理呢?

A: 视同普通的回写

总结

- 流水线设计的复杂性在于对冲突的覆盖性分析
 - ◆ 覆盖性分析使得设计与测试均具备了完整的正向设计的理论基础
 - ◆ 缺乏覆盖性分析，就不能断言是否处理了所有冲突
 - ◆ 避免了频繁的、无谓的试错；提高开发效率，确保开发正确性
- 教科书的不足
 - ◆ 没有覆盖性分析，难以满足大规模指令集的流水线设计与测试需求
 - ◆ 没有覆盖性分析，必然遗漏部分数据相关
 - 如lw~sw指令，必须暂停。但事实上可以通过增加转发MUX实现不停顿
 - 如cal~sw指令，未明确指出处理机制
 - ◆ RF内部的数据转发语焉不详
 - 内部转发：当读和写同一个寄存器时，读出的数据应该为要写入的数据

策略矩阵制导构造暂停案例

当建立策略矩阵后，可以反向构造暂停和转发案例

示例：rs策略矩阵制导的rs相关暂停用例

rs策略矩阵

<div>T_{new} T_{use}</div>	E			M			W		
	ALU	DM	PC	ALU	DM	PC	ALU	DM	PC
	1	2	0	0	1	0	0	0	0
0	S1	S2	F	F	S3	F	F	F	F
1	F	S4	F	F	F	F	F	F	F

TIPS

load类、store类、运算类、b类均包含多条指令。

因此，从指令角度，造成暂停的指令组合数量巨大。

rs相关暂停用例

S1	运算类 \$1, \$x, \$y b类 \$1, \$x, im	运算类 \$1, \$x, \$y jr \$1	
S2	load类 \$1, x(\$y) b类 \$1, \$x, im	load类 \$1, x(\$y) jr \$1	
S3	load类 \$1, x(\$y) XXXXX b类 \$1, \$x, im	load类 \$1, x(\$y) XXXXX jr \$1	
S4	load类 \$1, x(\$y) 运算类 \$x, \$1, \$y	load类 \$1, x(\$y) load类 \$x, y(\$1)	load类 \$1, x(\$y) store类 \$x, x(\$1)

策略矩阵制导构造转发案例

- 示例：rt策略矩阵制导的可以转发的rt相关用例

rt策略
矩阵

T _{use} \ T _{new}	E			M			W		
	ALU	DM	PC	ALU	DM	PC	ALU	DM	PC
	1	2	0	0	1	0	0	0	0
0	S	S	F	F	S	F	F3	F3	F3
1	F	S	F	F	F	F	F3	F3	F3
2	F	F1	F	F	F2	F	F3	F3	F3

TIPS

相对于转发类相关数据更为惊人。

覆盖性分析方法带来很多重要启示，进一步深化流水线认识。

rt相关
转发
用例

F1	load类 \$1, \$x, \$y store类 \$1, x(\$y)	启示：W级应该有向M级的转发通路
F2	load类 \$1, x(\$y) XXXXX store类 \$1, x(\$y)	启示：同步信息的流水线寄存器也是需求点
F3	运算类 \$1, \$x, \$y XXXXX XXXXX store类 \$1, x(\$y)	启示：RF需要支撑内部转发（2017新方法也可以采用外部显式转发）



与技术相关的注意事项

- 延迟槽：如果实现延迟槽，则对于如jal/b类等，为了使得PC+8能传递至W级，就必须在某个流水段再增加“PC+4”的功能
 - ◆ 1) 可以单独用一个adder实现
 - ◆ 2) 也可以在NPC中实现
- 目前的转发旁路综合方法以及转发控制，所有的转发均是“显式”实现的
 - ◆ 换言之，RF可以不具有内部转发功能
- 特别提醒：PPT中的各建模表格、流水线通路等，可用于设计工作的参考基础，但不能被简单的认为就是设计本身。
 - ◆ 图：多为示意性；表格：部分表格甚至被故意去除了部分内容
 - ◆ 好的学习方法：按照方法，自行推演出全部的设计过程

与技术相关的注意事项

□ $T_{\text{new}}/T_{\text{use}}$ 表格与表达式的关系

- ◆ 控制器中并没有一张表格；控制器中只有表达式（或者说是电路）
- ◆ 为使得流水线能正确处理所有数据冒险，设计师需要一种能覆盖所有数据冒险的分析方法
- ◆ 表格：是一种服务于覆盖性分析的形式建模方法，用于帮助设计师100%正确的构造表达式
 - 兼顾抽象与直观表达；建模速度快；能确保100%覆盖率
- ◆ 从表格到表达式：就是从设计到实现

与认识相关的注意事项

- 流水线开发复杂度远高于单周期和多周期，其主要原因在于并行性导致的思考点总量及关联度急剧攀升
 - ◆ 指令集的任何变化均会产生大量连锁反应
 - ◆ 类比：如果单周期、多周期是树，那么流水线就是图
- 除了流水线方法，确保流水线开发效率与正确性的重要因素之一：严谨的设计过程
 - ◆ 设计过程必须是显式的，而非隐式的
 - 显式的设计过程：各个设计环节必须有存证（文字、图、表、伪代码等）
- 只有详尽的显式设计过程，才能确保：
 - ◆ 正确并高效的完成设计调整与工程实现
 - ◆ 回溯整个设计过程以快速定位错误（这条甚为重要）

忠告

- ❑ 不要急于编码！
 - ❑ 设计越细致、越充分越好！
 - ❑ 所有的设计规划都会有回报的（是**加速型**回报）！
-
- ❑ 不要急于编码！
 - ❑ 设计越细致、越充分越好！
 - ❑ 所有的设计规划都会有回报的（是**加速型**回报）！
-
- ❑ 不要急于编码！
 - ❑ 设计越细致、越充分越好！
 - ❑ 所有的设计规划都会有回报的（是**加速型**回报）！



流水线实验的教学定位

- 流水线实验需耗费大量的时间与精力，有必要吗？！
 - ◆ 这是不少同学对此的质疑
 - ◆ 以下我浅谈一下流水线实验的设计初衷

- 现代工程由于复杂度非常高，故必须：
 - ◆ 采用基于模型和抽象描述的开发方法，否则难以精确描述开发对象
 - ◆ 建立严密的自顶向下、层次递进的设计过程，否则难以合理规划各环节
 - ◆ 设计过程必须是显式和严谨的，否则无法在高复杂度前提下完成增量设计和错误（故障）定位
 - ◆ 设计与实现必须分离，否则实现细节会严重干扰设计过程

流水线实验的教学定位

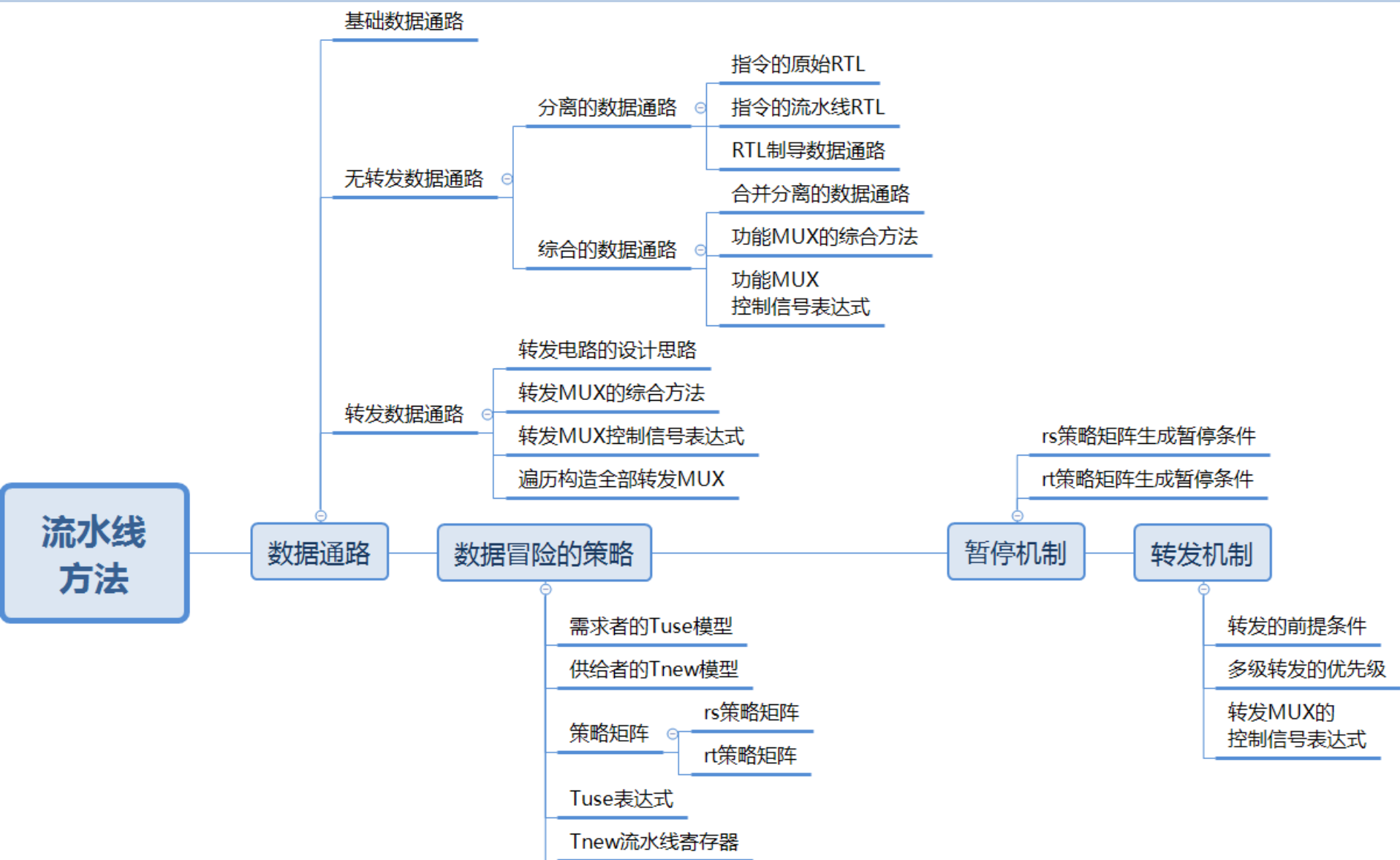
- ❑ 计算机专业是典型的工科专业。我们强烈期盼北航毕业生是未来的社会精英，是现代工程的领导者、驾驭者，而不能在未来面对现代工程时思考无绪、进退失据。
- ❑ 为此，我们需要寻找这样一个载体：首先，它能有效体现现代工程的特点；其次能依托这个载体来培养学生对现代工程开发的正确认识，理解并掌握其基本内涵与要点；最后也是最重要的是其复杂度是学生通过自身努力能够驾驭的。
- ❑ 经过多种分析、实践与权衡，我们认为全速5级流水线就是一个非常好的教学案例。
- ❑ 此外，我们还非常期盼能够借助这些实验进一步培养学生的抽象思维能力和严密的逻辑分析与推理能力。这些能力，也是社会精英所必需的基本条件之一。
- ❑ 以上就是流水线实验的设计思考。

流水线实验的教学定位

- “站在全系统的高度，建立系统的层次关系并厘清各层的组成要素及其作用关系，能对系统进行抽象建模并通过显式的逻辑推演完成设计过程，借助甚至开发相应工具来完成与加速设计到实现的转换”
 - ◆ 我将其称之为系统能力，并将相应的思维方法称之为系统思维
 - ◆ 这一能力并非只针对硬件系统有效，而是具有普适性

我非常希望包括流水线实验在内的计组实验，能有助于你们掌握这样的思维方法、具备这样的能力，是你们成为未来精英的助推器。

流水线设计的思维导图



新版本修改的注意事项

- 从设计的角度，需要一个简单的方法能发现所有可能的阻塞组合
- 对Tnew/Tuse的解读要体现出：
 - ◆ 1、如何分析
 - 发现暂停的思路：将D级与其后各级比较2个时间的早晚
 - 这样就将数据相关的分析表示为一个简单的数学模型
 - ◆ 2、表格与表达式的关系
 - 表格：给人看表格很容易理解模型
 - 表达式：硬件只能用表达式，因此就必须为每行、每列都构造变量；交叉点位“s”就对应行列变量的“与”

新版本修改的注意事项

❑ 暴力转发的最少控制信息分析

- ◆ 1、R编号==W编号； 2、W编号的真实性； 3、同级多个转发源的选择
 - 条件1：很好理解
 - 条件2：防止后级指令的相应域（[20:16]、[15:11]）与R编号巧合
 - 条件3：例如M级，包含ALU和PC4； W级包含ALU、DM、PC4
- ◆ 能否找到

❑ A1/V1， A2/V2：伴生关系

❑ 先总结冒险分析的基本思路

❑ 先总结数据相关的基本思路